Avaya VTCPD Features User Manual (Software Release 2.1 on Avaya MPS 2.1)

Avaya Business Communications Manager
Release 6.0

Document Status: **Standard**

Document Number: **P0602483**

Document Version: **03.41**

Date: **June 2010**

© 2010 Avaya Inc.
All Rights Reserved.

Notices
While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer
Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer
Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation(s) provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty
Avaya provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available to Avaya customers and other parties through the Avaya Support Web site: http://www.avaya.com/support
Please note that if you acquired the product from an authorized reseller, the warranty is provided to you by said reseller and not by Avaya.

Licenses
THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER, AND AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Copyright
Except where expressly stated otherwise, no use should be made of the Documentation(s) and Product(s) provided by Avaya. All content in this documentation(s) and the product(s) provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third Party Components
Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://support.avaya.com/Copyright.

Trademarks
The trademarks, logos and service marks ("Marks") displayed in this site, the documentation(s) and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the documentation(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party. Avaya is a registered trademark of Avaya Inc. All non-Avaya trademarks are the property of their respective owners.

Downloading documents
For the most current versions of documentation, see the Avaya Support. Web site: http://www.avaya.com/support

Contact Avaya Support
Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com/support

Table of contents

Chapter 1 — Preface 7

Scope 8
Intended Audience 8
How to Use This Manual 8
Organization of This Manual 9
Conventions Used in This Manual 10
Solaris and Windows 2000 Conventions 11
Trademark Conventions 11

Chapter 2 — Avaya VTCPD Overview 13

Host Communications Overview 14
MPS Software Architecture 16
Avaya VTCPD Features 18
Connection Types 19
Single Host Connection 19
One Connection Per Line 19
Multiple Hosts 19
Client and Server Modes 19
Application-Defined Mode 20
UDP Mode 20

Chapter 3 — Avaya VTCPD Configuration and Options 21

Configuration 22
The `services` File 22
Avaya VTCPD Daemon Names 22
Avaya VTCPD Port Assignments 22
Options Overview 24
Connection Options 26
Host Connection Options 26
Client Mode 26
Server Mode 27
Single Host Connection 28
Multiple Hosts Connections 28
One Connection Per Line 28
Multiple Avaya VTCPD Daemons 28
Application-Defined Mode 29
Dynamic Connections For Each Line 31
UDP Mode 33
Attaching to VMST 34
Application Connection Options 35
Application-Host Options 36
Monitoring Host Connections 38
Backup LAN 38

Time-Outs 39

Chapter 4 — VTCPD Messages 45

Message Format 46
Message Identification (ID) 50
Format of Outgoing and Incoming Messages 52
Message Routing 53
ISSUE-SEND-RECEIVE (ISR) 54
Reserving a Slot 57
Queuing Requests 57
Clearing the Connection 58
Changing Data Length 59
Application's Connection Choice 59
Replies Routing 60
Asynchronous Replies and Reply Notification 61
Unidentified Host Data 62
Administrative Line 64
ISSUE-RECEIVE-SEND (IRS) 64
Registering the ID 64
Inverted Mode 64
Application Connection Choice 66
Choosing the Daemon 67
Releasing Connection Slot and Handling Exceptions 69

Chapter 5 — VTCPD Application Programming 71

Communicating With the Host 72
Soliciting Host Data 73
Host Driven Applications 76
Receiving Data From a Known ID 77
Receiving the Next Available Host Message 78
Establishing or Changing the Host Specifications 79
Specifying the Host Connection 81
Specifying the Connection From the Application 81
Allowing VTCPD to Specify the Connection 82
Sending Messages 84
Routing Replies in a Host Driven Application 85
Variable Length Messages 86
Receiving Messages 87
Receiving Message Headers 88
Host Responses From Any Connection 89
Asynchronous Replies and Reply Notification 89
Retrieving Unidentified Host Data 89
Configurations That Require Multiple VTCPD 91
Specifying Which VTCPD 91
Using the VTCPD Port Number 91
Freeing the Host Connection 95
VTCPD Message Timing 96
Setting Resource Timers from an Application 97
Handling Resource Timeouts 97

Chapter 6 — VTCPD Debugging and Maintenance 101

Debugging 102
Displaying Debugging Information 102
Host Simulation 103
Fault Tolerance 104
VTCPD Status and Exception Conditions Reference 106
VTCPD Troubleshooting 107
Typical Problems 107
Questions and Answers 109

Chapter 7 — Index 111

This page has been intentionally left blank.

Preface

Scope

The *Avaya Voice Transmission Control Protocol Daemon (Avaya VTCPD)* Features User Manual documents the use of Avaya VTCPD. This manual provides background information and details about Avaya VTCPD configuration parameters and commands. This manual does not explain general telephony or host computer communications concepts.

See the *Avaya Media Processing Server Series COMMGR Reference Manual f*or details about general host-related configuration and operations commands. See the *Avaya PeriProducer Reference Manual* for details about application programming. For a list of user manuals use the *Avaya Reference Material* link in PeriDoc.

Intended Audience

This manual is intended for the staff members who configure and program the Avaya VTCPD. The reader should be familiar with telecommunications and computer equipment, their functions and the associated terminology. In addition, the reader must be familiar with the characteristics of the specific installation, including onsite power systems, computers, peripherals, and telephony components.

This manual assumes that the user has completed an onsite system familiarization training program conducted as part of the initial system installation. Basic knowledge of the Solaris or Windows 2000 operating system(s) is also assumed. In addition, they should be familiar with other site-specific operating procedures relating to the MPS that are due to specific application functions performed by the MPS and with any other equipment to which the MPS may be connected.

How to Use This Manual

This manual uses many standard terms relating to computer system and software application functions. However, for terminology that can only be explained in the context of the MPS system refer to the *Glossary of Avaya Media Processing Server Series Terminology* for specific term definitions.

Initially, read this manual at least once, from start to finish. Later, use the Table of Contents to locate topics of interest for reference and review.

If you are reading this document online, use the cross-reference links (shown in blue) to quickly locate related topics. <LEFT> click once with your mouse while positioned with your cursor over the cross-reference link. Click on any point in a Table of Contents entry to move to that topic. Click on the page number of any Index entry to access that topic page. For additional related information, use the *Reference Material* link in PeriDoc. To familiarize yourself with various specialized textual references within the manual, see ***Conventions Used in This Manual*** on page 10.

Periphonics is now part of Avaya. The name *Periphonics,* and variations thereof, may appear in this manual where it is refers specifically to certain product names and commands, for example, a *PeriProducer* application, the *PERImps* package, the

`perirev` command.

Organization of This Manual

The following briefly outlines the structure of this manual:

Chapter 1—Avaya VTCPD Overview
Introduces the Avaya VTCPD host daemon as a configurable host resource that allows the MPS to communicate with multiple external hosts.

Chapter 2—Avaya VTCPD Configuration and Options
Details the commands and parameter settings for configuring the system for Avaya VTCPD operations.

Chapter 3—Avaya VTCPD Messages
Details the message format and parameter settings.

Chapter 4—Avaya VTCPD Application Programming
Explains the basic concepts related to application-host interaction. The various message command tags and their attributes are documented.

Chapter 5—Avaya VTCPD Debugging and Maintenance
Provides Debugging and Maintenance information.

Conventions Used in This Manual

This manual uses different fonts and symbols to differentiate between document elements and types of information. These conventions are summarized in the following table.

Conventions Used in This Manual

Notation	Description
Normal text	Normal text font is used for most of the document.
important term	The Italics font is used to introduce new terms, to highlight meaningful words or phrases, or to distinguish specific terms from nearby text.
`system command`	This font indicates a system command and/or its arguments. Such keywords are to be entered exactly as shown (i.e., users are not to fill in their own values).
`command`, **`condition`** and **`alarm`**	Command, Condition and Alarm references appear on the screen in bold text and reference the *Command Reference Manual*, the *Condition Reference Manual*, or the *Alarm Reference Manual.* Refer to these documents for detailed information about Commands, Conditions, and Alarms.
`file name / directory`	This font is used for highlighting the names of disk directories, files, and extensions for file names. It is also used to show displays on text-based screens (e.g., to show the contents of a file.)
on-screen field	This font is used for field labels, on-screen menu buttons, and action buttons.
<KEY NAME>	A term that appears within angled brackets denotes a terminal keyboard key, a telephone keypad button, or a system mouse button.
Book Reference	This font indicates the names of other publications referenced within the document.
cross reference	A cross reference appears on the screen in blue text. Click on the cross reference to access the referenced location. A cross reference that refers to a section name accesses the first page of that section.
	The Note icon identifies notes, important facts, and other keys to understanding.
	The Caution icon identifies procedures or events that require special attention. The icon indicates a warning that serious problems may arise if the stated instructions are improperly followed.
	The flying Window icon identifies procedures or events that apply to the Windows 2000 operating system only.[1]
	The Solaris icon identifies procedures or events that apply to the Solaris operating system only.[2]
`PERIPRO RESOURCE BLOCK`	This font indicates PeriProducer resource blocks.

1. Windows 2000 and the flying Window logo are trademarks or registered trademarks of the Microsoft Corp.
2. Solaris is a trademark or registered trademark of Sun Microsystems, Inc.

Solaris and Windows 2000 Conventions

This manual depicts examples (command line syntax, configuration files, and screen shots) in Solaris format. In certain instances Windows 2000 specific commands, procedures, or screen shots are shown where required. The following table lists examples of general operating system conventions to keep in mind when using this manual with either the Solaris or Windows operating system.

	Solaris	**Windows 2000**
Environment	`$MPSHOME`	`%MPSHOME%`
Paths	`$MPSHOME/common/etc`	`%MPSHOME%\common\etc`
Command	**`<command> &`**	**`start /b <command>`**

Trademark Conventions

The following trademark information is presented here and applies throughout for third party products discussed within this manual. Trademarking information is not repeated hereafter.

Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, Windows, Windows 2000, Internet Explorer, and the Flying Windows logo are either trademarks or registered trademarks of Microsoft Corporation.

Netscape® and the Netscape N® and Ship's Wheel® logos are registered trademarks of Netscape Communications Corporation in the U.S. and other countries. Netscape Navigator is also a trademark of Netscape Communications Corporation and may be registered outside the U.S.

This page has been intentionally left blank.

1

Avaya VTCPD Overview

This chapter covers:

1. **Host Communications Overview**
2. **Avaya MPS Software Architecture**
3. **Avaya VTCPD Features**
4. **Connection Types**

Host Communications Overview

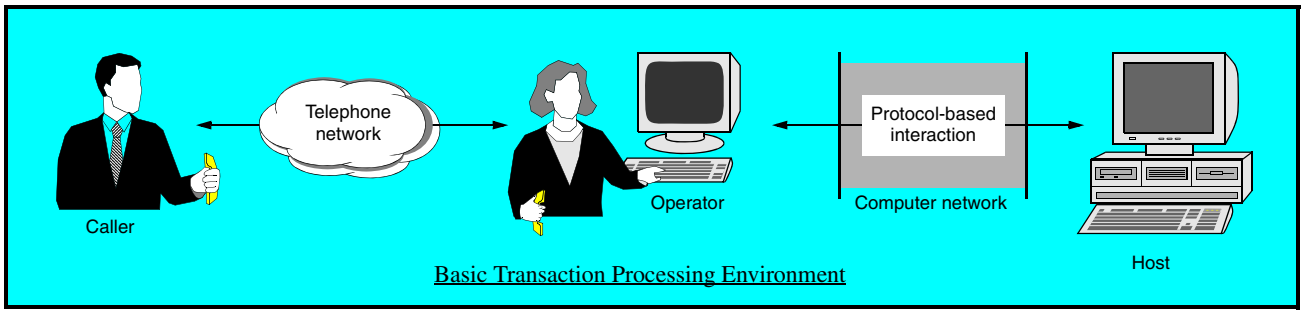
The Avaya Media Processing Server (Avaya MPS) Series product family is an Interactive Voice Response (IVR) system enhanced with multimedia and advanced telephone switching functions. The MPS can function as a standalone services system, with its own transaction processing and storage facilities, or it can be integrated into service-provider environments having their own central computer systems.

The *Avaya VTCPD* is an MPS software process used for integrating the system into a *Transmission Control Protocol/Internet Protocol (TCP/IP)* or *User Datagram Protocol (UDP)* network. Avaya VTCPD can accommodate a wide variety of applications and network configurations.

Avaya VTCPD is an alternative to using the MPS system's built-in host communications and protocol management facilities. The applications using Avaya VTCPD see it as a resource and access the Avaya VTCPD using the PeriProducer resource block. MPS applications that use Avaya VTCPD have to be developed specially for that purpose.

The MPS is the telephony services environment link between network features and the calling community. External systems connected to the MPS via the network are referred to as *host computers*. Generally, hosts are classified as mini, mainframe, and workstation. They provide database and transaction processing functions, which are integrated with the MPS voice and media features.

Before the advent of IVR systems, computer-based transactions involved having a live operator enter data and receive information through a terminal connected to a central computer system. IVR systems have automated this type of transaction.



Basic Transaction Processing Environment

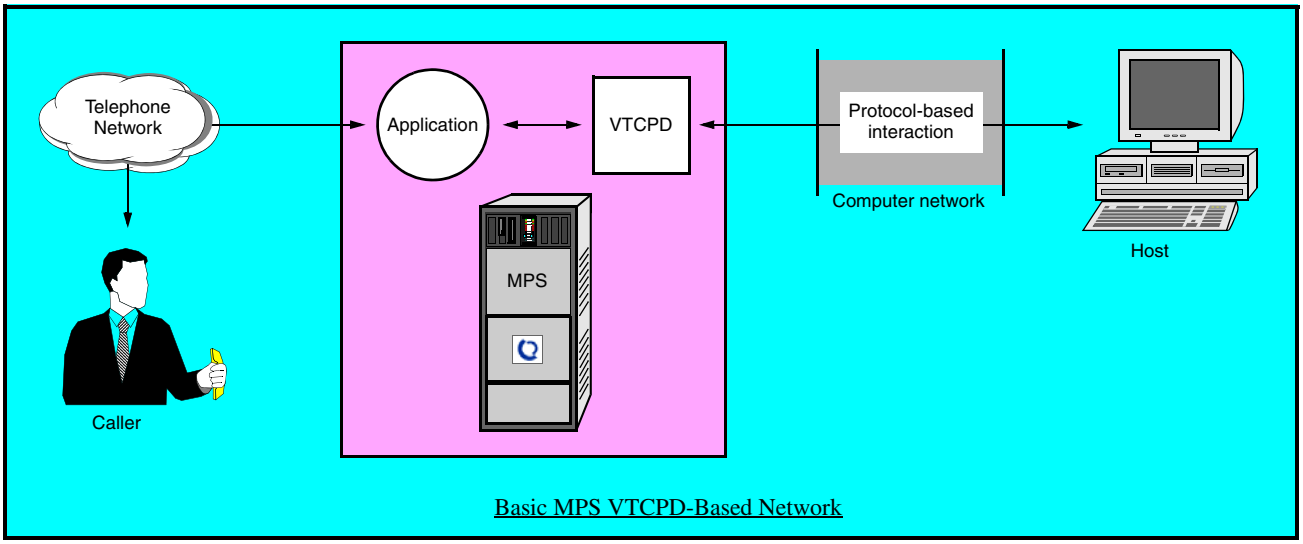
The *applications* control the actions of the MPS. They contain program instructions that tell the MPS how to perform functions, such as receiving caller input, providing voice output, and accessing the host. MPS applications are created using PeriProducer, which is a GUI-type editor that allows visual sequencing of application instructions. (See the *PeriProducer User s Guide*.)

Applications are directly associated with a specific MPS telephone line, or set of telephone lines. When a phone line is called, the application associated with that phone line activates and interacts with the caller and host based on the programmed instructions.

The application uses the Avaya VTCPD interface to provide the host with a communication format that matches the host's requirements. Read and write operations are performed between the application and the host, based on the characteristics of that particular host communications protocol.

A *protocol* is a standardized format for data transmitted between computer systems, which consists of command codes, data fields, and delimiters. that both computers can recognize. To send and receive data to/from a host using Avaya VTCPD, applications must be designed to format and decode messages according the protocol expected by the host.

An application with a Avaya VTCPD interface can communicate with several hosts, provided the hosts are using the same protocol. Applications can change the host session (switching from one host to the other) as needed.



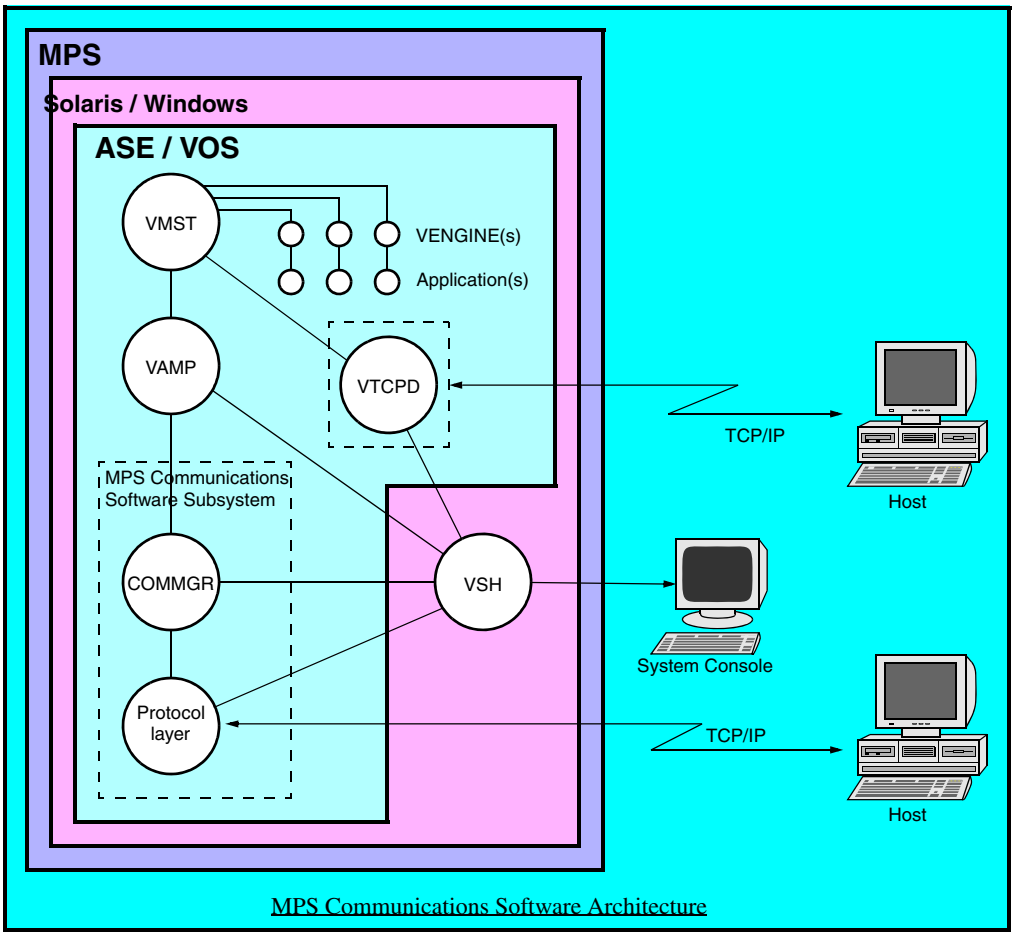
Basic MPS VTCPD-Based Network

MPS Software Architecture

The *MPS Communications Software Subsystem* is the MPS software specifically dedicated to host interaction. It resides in the MPS' *Voice Operating Software (VOS)*, and comprises the *Communications Manager (COMMGR)* process and *protocol layer.*

The COMMGR provides a protocol-independent interface between MPS applications and the protocol layer. The COMMGR also performs host-related phone line configuration, application-to-host session mapping, and host input/output message processing. Message interaction at the protocol level is handled by processes in the protocol layer. Together, the COMMGR and protocol layer manage most of the lower-level system functions related to host communications, and applications need only issue high-level send and receive commands to interact with a host.

Avaya VTCPD is an alternative to using the MPS Communications Software Subsystem (*COMMGR* process and the *protocol layer)*. Avaya VTCPD is intended for host environments that have existing TCP/IP or UDP software infrastructures that implement unique variations of standard protocols. MPS applications using Avaya VTCPD must be specifically coded to interact with the host(s) at the protocol level.



MPS Communications Software Architecture

Avaya VTCPD and the MPS Communications Subsystem are separate software entities. At any one time, an application uses one or the other to interact with a host. An MPS can be configured to use both software systems, and applications can switch back and forth as needed. For example, the MPS can be set up to use Avaya VTCPD to interact with one host and the communications subsystem to interact with another. (For additional information, see ***Connection Types*** on page 19.)

This manual documents only the Avaya VTCPD host interface. For more information about TCP/IP links, host communications, and protocols, refer to the *Avaya Media Processing Server Series COMMGR Reference Manual*.

System processes that are important from a host communications perspective are described below.

MPS System Software

VSH	*Vshell* is the command interface for MPS configuration and operations. Configuration and status commands can be entered from the VSH tool as needed. VSH also receives status information from the various system processes and displays messages on the console as appropriate.	
ASE	The *Application Service Environment* software is dedicated to providing the data and services requested by applications. The ASE exists on a separate workstation, referred to as the *applications processor*. The workstation can be either an open-systems Solaris or Windows 2000 implementation.	
	applications	Interactive Voice Response or multimedia script created with PeriProducer. An application runs on a system phone line. Multiple instances of the same application can be assigned to different lines.
	VENGINE	Software process that executes an application. A single VENGINE process is required for each application telephone line.
	VMST	The *VENGINE Message Server* provides a message funnel between the ASE and VOS processes. On a node that contains multiple MPS systems, VMST provides connectivity between the application processor and each MPS. One VMST is required for each MPS.
	VAMP	The *Voice Application Management Process* provides an interface between the PeriProducer 3.0 voice applications and the VOS subsystem.
VOS	The *Voice Operating Software* is the set of processes that provide the low-level system functions in the MPS. These functions, including telephony and host I/O, are common to most types of applications.	
	COMMGR	The *Communications Manager* provides a generic application interface for host communications services. A single COMMGR is required for each MPS in the network.
	protocol layer	One or more software processes that implement the particular communications protocol. The protocol layer links the COMMGR with the host network.

Avaya VTCPD Features

The *Avaya VTCPD* process facilitates communication between application programs (PeriProducer) and one or more external hosts.

The PeriProducer applications see the Avaya VTCPD daemon as a resource, with a configurable name. The applications use the PeriProducer resource blocks to provide the GET, FREE, SEND RESOURCE, and RECEIVE RESOURCE functions. The following table correlates specific VRAM code to PeriProducer blocks:

Table 1: VRAM Commands and PeriProducer Resource Blocks

VRAM Command	PeriProducer Resource Block
ISSUE GET	**GET**
ISSUE SET	**CONTROL**
ISSUE FREE	**FREE**
SEND RESOURCE	**SEND**
RECEIVE RESOURCE	**RECEIVE**

The external hosts see the Avaya VTCPD daemon as a configurable host resource and communicate through the daemon's TCP/IP or UDP connections.

The Avaya VTCPD daemon can connect with one or more external hosts and accommodates a wide variety of application requirements and host configurations. The daemon can also be used to monitor the elapsed time on any outstanding request and also provides a timing utility for applications.

One limitation is that a Avaya VTCPD daemon can run only one protocol at time. It can connect to several hosts using the same protocol, but if a new protocol is required another Avaya VTCPD process must be started for the new protocol.

Configurations Include:

- Connection to multiple VMST processes
- One or more connections to a single host
- Multiple connections to multiple hosts
- Connections to (yet-to-be-specified) hosts and port numbers
- Multiple Avaya VTCPD daemons with one or more hosts (one connection per line)

Avaya VTCPD can also be configured to automatically switch to a secondary host, in the event of a primary host failure.

This document only deals with the Avaya VTCPD daemon process. For information about application communication with TCP/IP hosts, refer to the *Avaya Media Processing Server Series Application Programming Reference Guide*.

Connection Types

The Avaya VTCPD process provides a variety of connection types to support diverse application requirements and host configurations. Avaya VTCPD can also be configured to automatically switch to a secondary host, in the event of a primary host failure.

The Avaya VTCPD daemon can run only one protocol at time. It can connect to several hosts using the same protocol, but if other protocols are required, additional Avaya VTCPD process must be started for each protocol.

Single Host Connection

The single host network is Avaya VTCPD basic configuration. The process connects to a single host using the host name and TCP/IP port number (specified as arguments on the command line). If a host name is not specified, Avaya VTCPD is configured as a server, in which case it accepts connections on the specified port.

An application uses the SEND function (PeriProducer resource block) to send a message to the host and it issues the RECEIVE function (PeriProducer resource block) when it expects a response.

One Connection Per Line

Avaya VTCPD can create as many connections to a host as there are lines available. In this type of configuration, each application has its own connection to the host. Avaya VTCPD can also be started with the number of links equal to the number of phone lines, without specifying host names and port numbers. This allows Avaya VTCPD to open a connection for each new call, as needed.

Multiple Hosts

Multiple Avaya VTCPD daemons can connect to multiple hosts, each handling different functions and protocols. Applications can switch the host sessions as needed.

Client and Server Modes

The Avaya VTCPD daemon is capable of running in both client and server modes. In client mode, Avaya VTCPD connects to a host as a client process. In the server mode, the Avaya VTCPD daemon accepts connections on the specified TCP/IP port. In a multiple-host environment, for different hosts, the daemon supports both client and server modes simultaneously.

Application-Defined Mode

When all or some host specifications are unavailable, the Avaya VTCPD daemon can be started using the Application-Defined mode. The links will remain unavailable for applications until port and host information are associated with the links (host information is needed only in client mode). However, the administrative applications can dynamically assign or change host specifications. The specific links are differentiated from other host links by the number (any additional host or port information is optional). The host name can be either the name or IP address.

UDP Mode

All Avaya VTCPD modes (client, server) also support User Datagram Protocol (UDP) -type host connections. The host must extract the Avaya VTCPD address and port number from the UDP message to enable it to reply to applications.

2

Avaya VTCPD Configuration and Options

This chapter covers:

1. **Configuration**
2. **Options Overview**
3. **Connection Options**
4. **Application-Host Options**

Configuration

The `services` File

The `services` file contains configuration information required by Avaya VTCPD and must be set up accordingly.

Information required in the `services` file includes:

- Avaya VTCPD daemon names
- Port assignments used by Avaya VTCPD

The `services` file contains a list of processes (i.e., the services), some of which can be accessed by call processing or by using administrative applications. This file defines the port and protocol associated with each service and is used by the Application Services Environment (ASE) software process group (see ***Communicating With the Host*** on page 72.) For Windows systems, the `services` file is stored in `%ASEHOME%\etc`. In Solaris systems, the file is stored in the `$ASEHOME/etc` file. Avaya VTCPD uses the `services` file definitions for the Avaya VTCPD Daemon Names, the Avaya VTCPD Port Assignments and the VMST Port Assignments.

Avaya VTCPD Daemon Names

If more than one Avaya VTCPD daemon is running simultaneously, then each Avaya VTCPD must have a unique name. The unique Avaya VTCPD names must be defined in the `services` file before they can be used by the Avaya VTCPD command line option **`-n`** (see ***Specifying Which VTCPD*** on page 91.)

In the example `services` file *(*page 23*)* ***vtcpd*** *and* ***newhost*** are VTCPD daemon names.

Avaya VTCPD Port Assignments

In a typical configuration, all Avaya VTCPD daemons must have different names and be distinguished by the unique port numbers. In the example `services` file *(*page 23*)* ***vtcpd*** *and* ***newhost*** are Avaya VTCPD daemon names.

`services` File Field Definitions

Variable	Description
`Service`	Specifies the process name.
`Port(s)`	Identifies the system ports from which each process is accessed. The port numbers represent internal handles to the VMST processes (i.e., they are not TCP/IP ports). The numbers must not exceed 254, must be unique in this file. If this file is changed, all instances of VMST and services must be restarted.

`services` **File Field Definitions**

Variable	Description
`Protocol`	Defines the protocol used when accessing each process.

Example: `services` **File**

```
# Service       Port(s)       Protocol
##############################
vmst            1-10
htmls           12-15         linfo
periq           16-17         linfo
.
.
.
vtcpd           132-145       linfo
newhost         150-151       linfo
.
.
.
```

Options Overview

The Avaya VTCPD options support a wide range of application requirements and host configurations. The options list provides a quick reference to the available options and the options examples provide a quick overview of how some of the options can be used. The ***Connection Options*** on page 26 and ***Application-Host Options*** on page 36 sections provide in-depth discussions of those option types. These Avaya VTCPD options are specified as command line arguments.

Option Type	Option	Description
Message Processing	**-l [#:][host:]port #**	Links to host machine: TCP/IP port.
	-f len | d:delim	Message length | delimiter, where the delimiter can be a character or a string.
	-f m:maxlen	Maximum message length, applied in addition to other **-f** options.
	-f l:flen[:pos[:hdr]]	Message length field [position [header length]].
	-f a:flen[:pos[:hdr]]	Message length field in ASCII [position [header length]].
	-F opts	Same as **-f**, but for **incoming** messages only.
	-i{len:pos|len@str}	ID length:pos (from 1) or length@idstring.
	-j opts	Same as **-i**, but for **incoming** messages only.
	-I{len:pos|len@str}	Set ID, length:pos (1, 2, 3, or 6 length) or length@idstring.
	-P -|#	Common pool size (- unlimited [999], default).
	-p #	Per line pool size.
	-q #	Queue requests per link (when there are # unanswered by host requests).
	-Q #	Queue requests in global queue (when all links have # unanswered requests).
	-d sec	Connection timeout (if a request is not answered). Use with **-q**/**-Q** only.
	-D sec	Connection timeout (if a request is not answered). Close the link. Use with **-q**/**-Q** only.
	-a line[:link]	Administrative line for unidentified host data.
	-A line[:link]	Administrative line for all host data.
	-m {a|n|i|t|u|U|h}	Mode {send async|notify|invert|truncate spaces|send **hostup** cond|UDP|keep the header}.
	-m{l|L|I|R}	Mode {use same link|keep link information|ISSUE doesn't reserve a slot|regular round robin}.
	-mB	Binary headers are in non-network order.
	-u port	Port to listen on for server's replies (in UDP client mode only).

Option Type	Option	Description
Connection to VMST	**-v [node:]mps**	MPS number with optional remote node name.
	-s #	Port number to VMST.
	-n name	Optional service name (default Avaya VTCPD).
Debugging Options	**-X {a|2}**	Debugging level (a- display all messages to/from VMST, 2-host).
	-r device	Redirect output (to /dev/tty#,/dev/console).
	-H	Help.
Examples:	**vtcpd -v1 -s160 -l 4:servh:10000 -f d:0x0a -i 48:6** Specifies four connections, unlimited pool, NL message delimiter, and 48 bytes message ID from 6th position.	
	vtcpd -v1 -s160 -l 2:servh:9000 -m d -q 3 -P 20... Specifies two connections, a reply can arrive from a different connection, 20 slots are allocated in the global pool, and requests are queued for up to 3 seconds.	
	vtcpd -n mainmps -v1 -s160 -l48:servh:9000 -p 1 -f 65 -ma VMST is running on the machine 'mainmps'. There is one connection per line (no ID), fixed-length messages and asynchronous host data delivery.	
	vtcpd -v1 -s160 ... -f d:'<end>' -f m:64000 ... Sets the message delimiter to the string "<end>" and the max message length to 64000 bytes.	

Connection Options

The Avaya VTCPD daemon can run as a client or server. In a multiple-host environment, for different hosts, the daemon supports both client and server modes simultaneously, provided unique port numbers are specified for each mode. In client mode, Avaya VTCPD connects to a host as a client process. In the server mode, the Avaya VTCPD daemon accepts connections on the specified TCP/IP port.

Host Connection Options

Host connections are specified by the Avaya VTCPD daemon's command line options. Each connection is defined by the specified host machine name (or IP address) and the TCP/IP port number.

The **-l** command option is used to specify whether to run the Avaya VTCPD daemon as a client or server (see ***Client Mode*** on page 26 and ***Server Mode*** on page 27). In a multiple-host environment, for different hosts, the daemon can support both client and server modes simultaneously, if appropriate (and different) port numbers are specified.

Client Mode

In this mode, Avaya VTCPD connects to a host as a client process. The **-l** option is used to specify the number of connections (links) to the host using a specific TCP/IP port. This option can be used multiple times.

The following table shows the Avaya VTCPD client mode syntax.

Client Mode Connection Options

vtcpd	**-l [#:][host]:[port]**	
Args:	**#**	Indicates the number of connections to the specified host. (The default is **1**.)
	host	Indicates the name or IP address of each host.
	port	Indicates the TCP/IP port number. (Port numbers 7000-7256 are reserved for the VMST and must not be used.)
Examples:	**vtcpd -l 2:eagle:10000 -l eagle:10001 -l hawk:11000** Specifies three connections to the host name *eagle* (TCP/IP ports 10000 and 10001) and one connection to the host named *hawk* (port 11000).	
	vtcpd -l 1::5000 Specifies a link to a yet-to-be-defined host via port 5000.	
	vtcpd -l 2:eagle: Specifies two links to the host *eagle* via a yet-to-be-defined port.	
	vtcpd -l 3:: Specifies three yet-to-be-defined links.	

Client Mode Connection Options

vtcpd	**-l [#:][host]:[port]**
	In client mode, if the host or port arguments are omitted, the links are unavailable to applications until this information is supplied by an administrative application. This allows the links to be allocated dynamically. See the *Avaya Media Processing Server Series Application Programming Reference Guide* for details.
	Ports 7000-7256 are reserved for VMST and must not be used.
	Correct sequencing of the connection specifications is important because applications can refer to connections by numeric designation when requesting a specific connection. The connections are numbered in the order specified on the command line. (For example, the numeric designation of the *hawk* connection as shown above is 4.)
	If the number of connections is not specified, it defaults to one.

Server Mode

In the server mode, the Avaya VTCPD daemon accepts connections on the specified TCP/IP port. The server mode uses the **-l** option and the specified port number to configure the connections between the Avaya VTCPD daemon and the TCP/IP port.

The following table shows the Avaya VTCPD server mode syntax.

Server Mode Connection Options

vtcpd	**-l [#:][port]**	
Args:	**#: [port]**	Indicates how many connections can be accepted on a given port. The daemon can support client and server modes simultaneously for different hosts (in which case, the specified port numbers must be different).
Example:	**vtcpd -l 3:** Specifies three links in server mode on yet-to-be defined ports	
	If all connections are in server mode, the switch to the backup LAN (see ***Backup LAN*** on page 38) will not be reversed even when the LAN is down.	
	The Avaya VTCPD daemon always uses the server mode if a host name is not specified using the **-l** option.	
	If the port argument is omitted, the links are unavailable to applications until this information is supplied by an administrative application. This allows the links to be allocated dynamically. See the *Avaya Media Processing Server Series Application Programming Reference Guide* for details.	

Single Host Connection

The Avaya VTCPD daemon connects to a single host using the host name and TCP/IP port number (specified as arguments on the command line). If a host name is not specified, Avaya VTCPD is configured as a server, in which case it accepts connections on the specified port.

The application uses the SEND function (PeriProducer resource block) to send a message to the host and the application issues the RECEIVE function (PeriProducer resource block) when it is expecting a response.

Multiple Hosts Connections

The Avaya VTCPD daemon can be connected simultaneously to several hosts, provided each host is specified (on the command line) when the daemon is started. These connections are made with no priority given to any particular host. Avaya VTCPD sends messages on a round-robin, load-balancing basis. The round robin system is a form of non-sequential access, in which the system maintains a list of available Virtual Terminals (VT) and assigns the first VT in the list to the requesting line. When the VT is freed, it is put to the end of the list. The nonsequential method uses all the VT in the list equally.

The Avaya VTCPD assumes that each host provides equivalent services. The application can override Avaya VTCPD and choose a specific host. The number of outstanding requests on a given connection can be limited (if required by the host). If Avaya VTCPD cannot send a message to a host because the specified limit is exceeded, it returns an appropriate condition message to the application.

One Connection Per Line

Avaya VTCPD can create as many connections to a host as there are lines available. In this type of configuration, each application has its own host connection. Message IDs are not required (in this type of configuration) because there is only one connection per line.

Multiple Avaya VTCPD Daemons

Multiple Avaya VTCPD daemons can be connected to one or more hosts, each handling different message formats and services. The application uses the resource names to identify the messages (see ***Specifying Which VTCPD*** on page 91.)

Application-Defined Mode

The Avaya VTCPD daemon can start when all or some host specifications are unavailable. The links will remain unavailable for applications until port and host information is associated with the links (host information is needed only in client mode). However, the administrative applications can dynamically assign or change host specifications. The specific links are differentiated from other host links by the number (any additional host or port information is optional). The host name can be either the name or IP address.

To define such hosts use the option:

```
-l #:hostname:port
```

Using this option **`hostname`** or **`port`** can be omitted:

Examples:

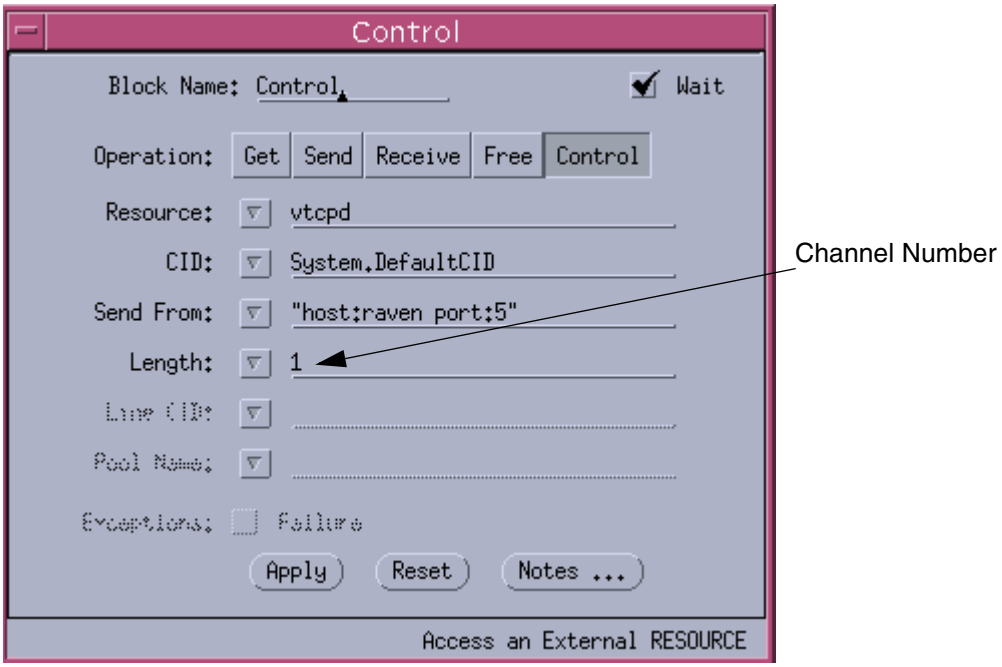
`-l 1::5000`
Specifies a link to undefined host through port 5000

`-l 2:ablaze:`
Specifies 2 links to host 'ablaze' through the as yet undefined port

`-l 3::`
Specifies 3 completely undefined links

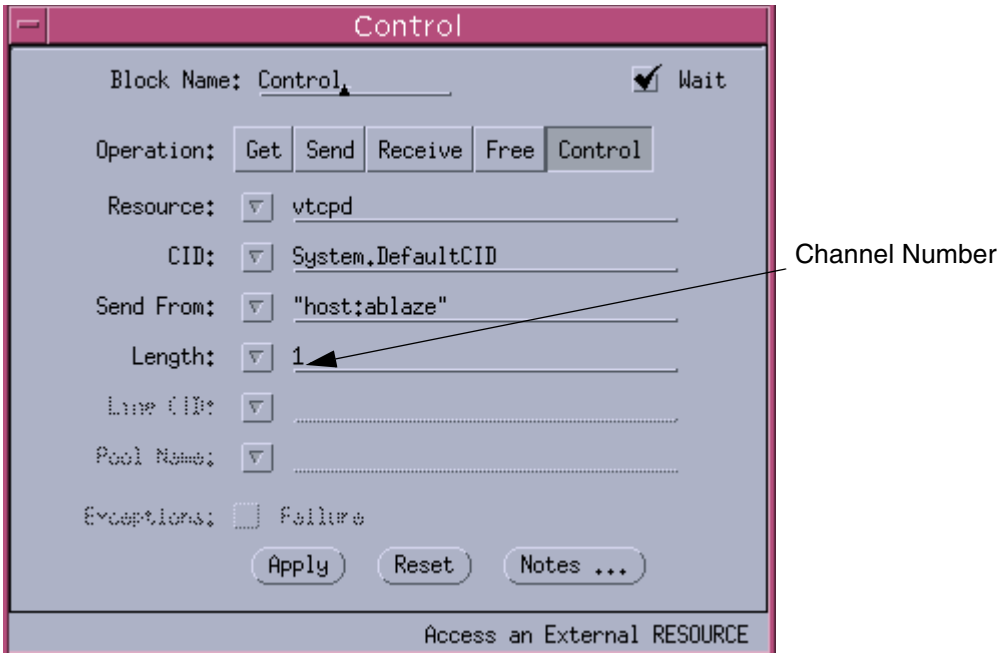
`-l 3:`
Specifies 3 links in server mode on yet to be defined ports

The links are unavailable for applications until port and host information is associated with the links (in client mode, only host information is needed). Administrative applications can dynamically assign or change host specifications, as depicted by the following sample PeriProducer screenshot:

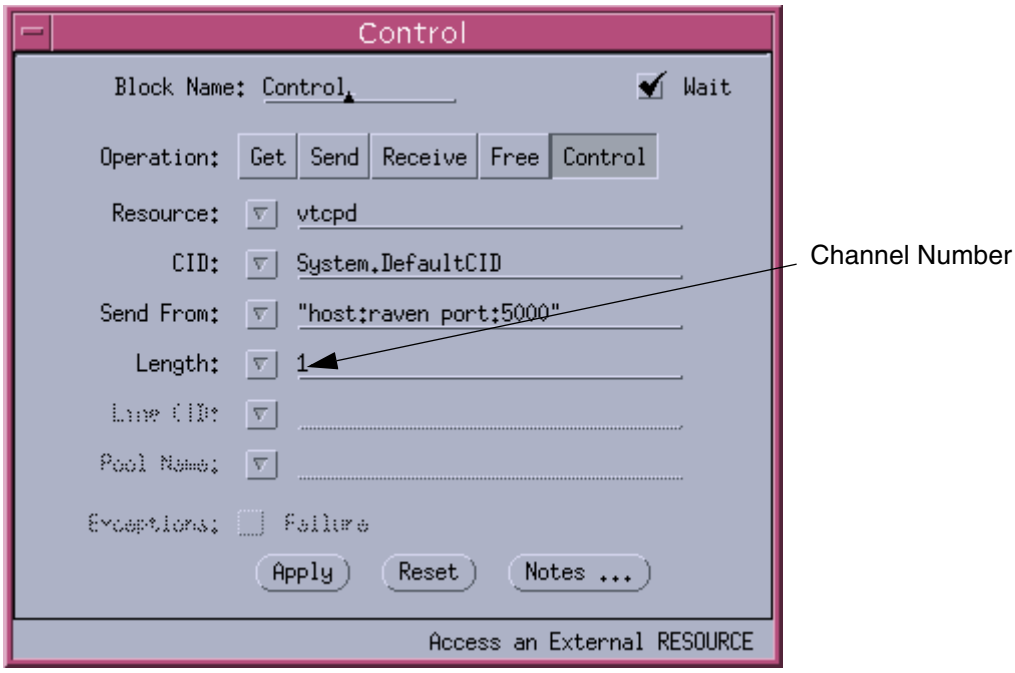


A specific host link can be differentiated from other host links by the number (any additional host or port information is optional). The `hname` uses either the name or IP address: host:192.84.160.127.

For example, for the first link in the examples above, set the host using:



For the server type connections, set the assignment using:



If CHANNEL is specified and the corresponding link is closed first, then it is connected.

If CHANNEL is not specified, the assignment is applied to all undefined links (links missing some host specifications).

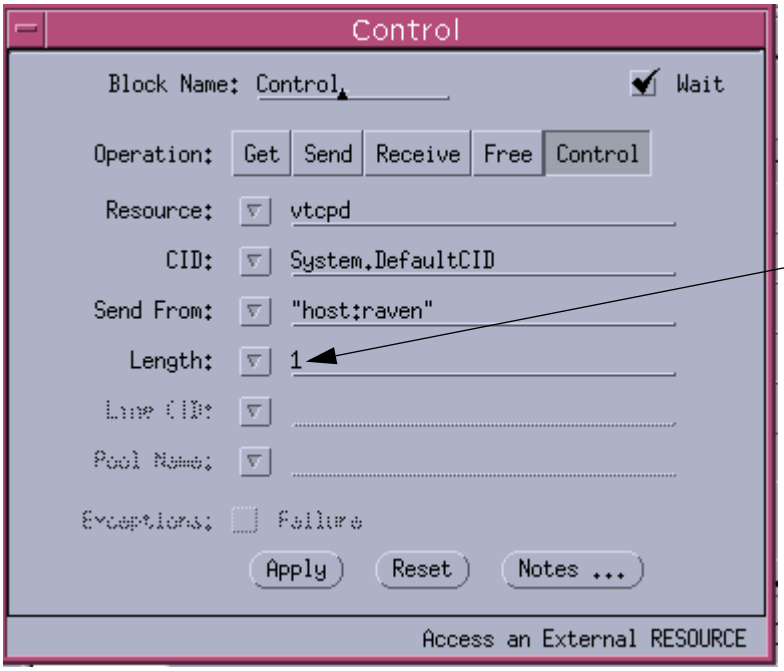
The **CONTROL** operation of the PeriProducer resource block forces the daemon to re-establish a connection (for example, when an administrative application misses pings from the host).

Dynamic Connections For Each Line

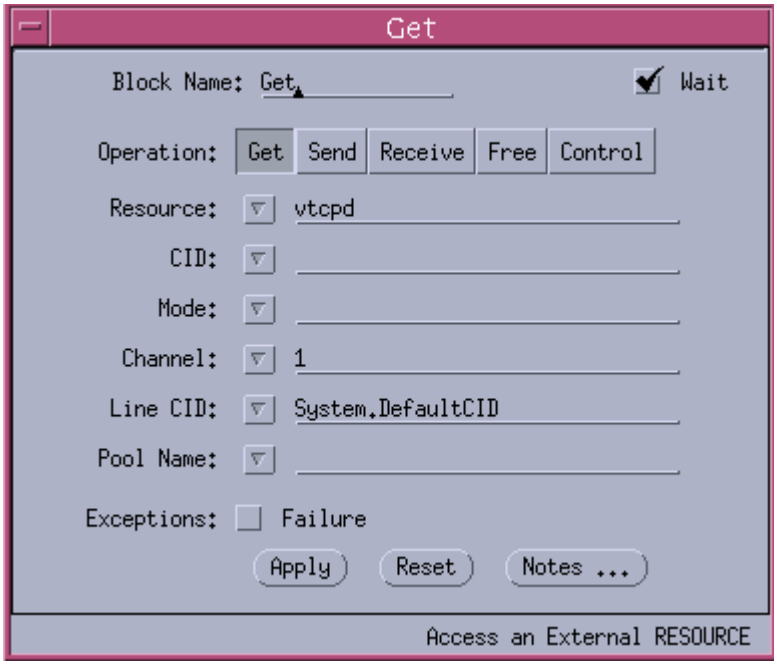
To allow the Avaya VTCPD to open a new connection for each new call, start Avaya VTCPD with the number of links equal to the number of phone lines and do not specify the host name or port number. This also tells the host to expect a new connection for each line (and each call). Do not specify the host name or port number, since the host name or port number specifications will prevent Avaya VTCPD from opening real connections. For example:

```
vtcpd -l 48::10000 ...
```

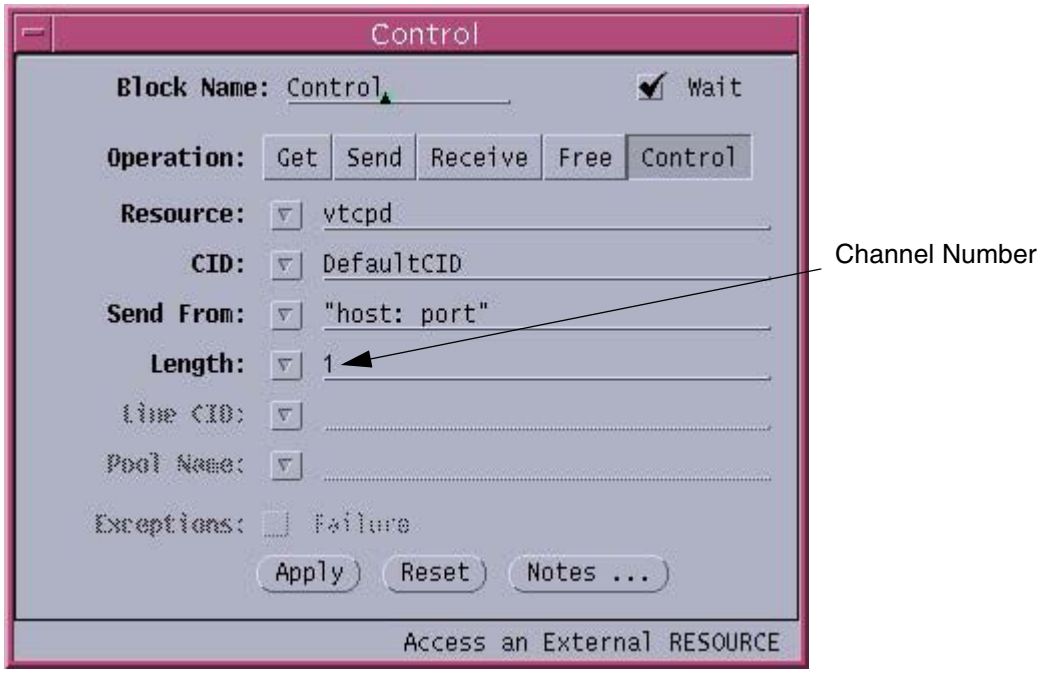
At the beginning of each call, use the **CONTROL** operation in the PeriProducer resource block. This operation closes the previous connection and opens a new connection.



Next, use the **GET** operation. Allow time for the Avaya VTCPD to establish the connection:



To close the connection, use:



If the host and port are not specified, it means that the connection is closed.

UDP Mode

The daemon can run in the UDP mode and the other Avaya VTCPD modes (client, server) will support UDP-type host connections. The host must extract the Avaya VTCPD's address and port number from the UDP message to be able to reply to the applications. Normally, the Avaya VTCPD in a client mode uses an ephemeral port number, obtained from the kernel. This forces the daemon to use the specified port to listen to replies from the server. The option has no effect in a server mode or in non-UDP mode.

The daemon can run in UDP mode, (specified by the **`-m U`** command line option).

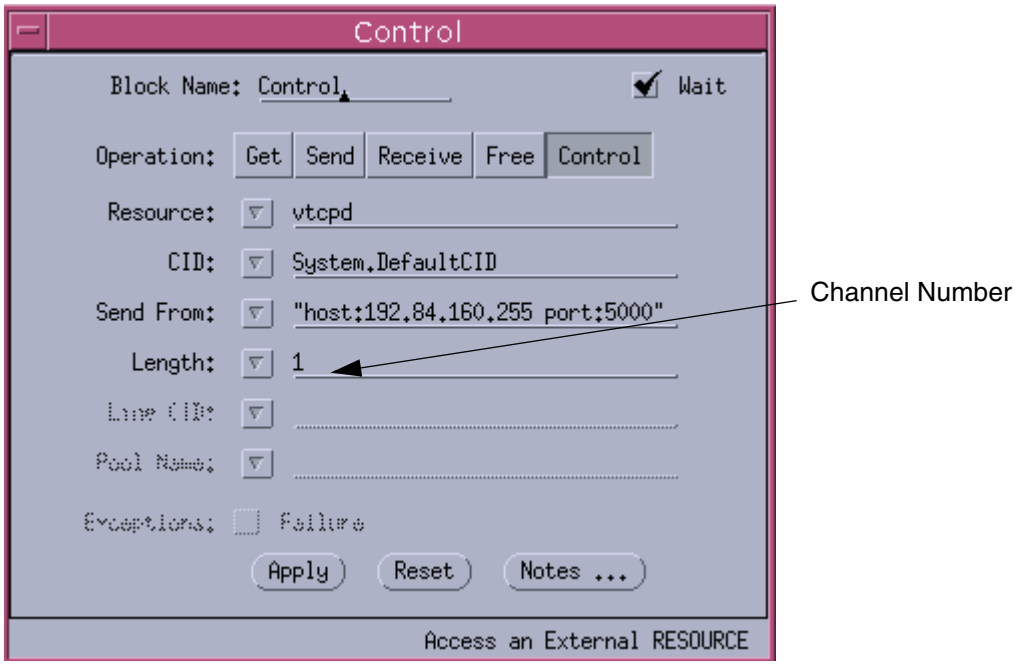
```
vtcpd ... -u port ...
```

This option forces the daemon to use the specified port to listen to replies from the server. The option has no effect in a server mode or in non-UDP mode.

In the UDP mode, the daemon typically cannot determine when the host goes down. Consequently, when the host is down, the application does not usually receive a **`hostdown`** condition. However, the application can receive a **`hostdown`** condition if the daemon receives a corrupted message from the host. In the UDP mode, an application can force the daemon to broadcast on the [local] network by specifying a proper IP address in the SET command. In the UDP mode, the whole message comes

from a host in a single package.

The following figure is an example of UDP mode.



This command broadcasts on subnet 192.84.160.* and port 5000.

In the UDP mode, the whole message comes from a host in a single package, which means that the Avaya VTCPD option **`-f`** should not be used.

Attaching to VMST

Avaya VTCPD can attach to any specified instance of VMST running on a LAN-based host node. If the target VMST resides on another machine, the name of the target node must be specified on the command line. To attach to VMST, Avaya VTCPD has to use one of the service ports specified in the `services` file (see the *Avaya Media Processing Server Series System Reference Manual*).

If there is a need to process messages with different formats, several Avaya VTCPD daemons can be attached to the same VMST. Each VMST must have a unique name (defined in the `$ASEHOME/etc/services` file). Applications address a particular instance of VMST by its defined name. In this case, each VMST must have a unique name specified on the command line option **`-n name`**. The daemon attaches to the vmst specified using the command line option; **`-v [node:]number`** (for example, **`-v 25 or -v mpssp:12`**).

To attach to the VMST, the Avaya VTCPD daemon has to use one of the Avaya VTCPD service ports specified in the `$ASEHOME/etc/services` file. The specific port is defined by the **`-s port#`** option.

Attaching to VMST

`vtcpd`	**`[-v {<name>|<#>}] [-N <host>] -s <port> -n <VTCPD> [-m u]`**	
Args:	**`-v {<name>|<#>}`**	Identifies a particular VMST to which Avaya VTCPD establishes a connection. If the VMST resides on another host node, the number of the MPS must be specified.
	`-N <host>`	Specifies that the target VMST resides on the indicated host node.
	`-s <port>`	Specifies the port number assigned to the target VMST. This must be a valid VMST port as defined in the `services` file.
	`-n <VTCPD>`	Defines a name used by applications to identify this instance of Avaya VTCPD. This is used when there are multiple instances of the Avaya VTCPD process, and applications need to distinguish between them. The names have to be specified in the `services` file. VMST has to be restarted to reread this file.
Example:	**`vtcpd -v 25 -N mainnode -s 160 -n hserver1`** Starts the VTCPD daemon attaching to the VMST on MPS number 25 (associated with the UNIX node called *mainnode*) via port 160. This VTCPD daemon is assigned the name *hserver1*.	

If the **`-v`** option is not specified the daemon attaches to all VMST instances running on the same host node. This configuration can create a bottleneck situation and should be used with caution if the number of available host links is limited.

Application Connection Options

The following options are available to applications:

- The Avaya VTCPD daemon chooses the connection and the corresponding slot in the pool using round robin balancing among slots with the minimum number of outstanding requests (by default). If command line option **`-m R`** is specified regular round robin load balancing is selected. If all slots are already used, the request is rejected.

- An application specifies which host connection (link) to use. The daemon assigns the slot or rejects the request if the pool is full.

Application-Host Options

This document only provides an overview of application programming requirements for Avaya VTCPD communications. For a detailed discussion on developing applications for Avaya VTCPD functions, see the *Avaya Media Processing Server Series Application Programming Reference Guide*.

Typically, PeriProducer applications are programmed to perform the following steps in the indicated order using the functions of the Resource block:

- Acquire the Avaya VTCPD resource (**GET**)
- Send a message to the host (**SEND**)
- Receive a reply from the host (**RECEIVE**)
- Free the Avaya VTCPD resource (**FREE**)

To perform these functions Avaya VTCPD must be able to:

- reserve an available host connection (the slot)
- send data to the host (associating the Message ID field with the phone line)
- receive data from the host based on the specified Message ID
- release the slot.

When Avaya VTCPD acquires a slot, the it sends the *gotres* condition to the application. Data associated with the *gotres* condition contains the connection number, slot number, and Avaya VTCPD port number in the format `XXX:YYY:ZZZ`.

If a reply from the host arrives before the application issues the **RECEIVE** command, the reply is held by Avaya VTCPD daemon (by default). If the application terminates, the slot is released automatically.

The **SEND** request includes only the data portion of the message, because the delimiter (**-f d**) or length header (**-f l**) is provided by Avaya VTCPD (by default).

After the initial **SEND** request, **SEND** and **RECEIVE** requests can be mixed in any order. The Message ID of the last **SEND** is in effect (until the slot is freed) and is used to identify consecutive **RECEIVE**s.

For example, in the following scenario, the application receives two messages with the specified Message ID-field:

- Acquire the Avaya VTCPD resource
- Send a message to the host
- Receive a reply from the host
- Receive a second reply from the host
- Free the Avaya VTCPD resource

During the interval between acquiring the resource (**GET**) and when the resource is released (**FREE**), the slot is reserved for that application line. If the size of the slot pool is limited, it might be necessary to reserve the slot for a shorter period of time. Using the **SEND** and/or **RECEIVE** requests releases the slot as soon as the host data arrives, freeing up the slot quicker than if the **GET**, **FREE** operation combination was used.

If permanent slot allocation is not desired, use the configuration option **-m I**. This allows the application to execute the **FREE** request to provide slot release and still permits dynamic slot allocation for each **SEND** and/or **RECEIVE**.

Application-Host Options

vtcpd	**[-m [d] [{a|n}] [{l|L}] [{I|R}]]**	
Args:	**-m d**	Allows a host reply to come from any connection. Normally, a reply arrives from the host through the connection used to send the request. Using this option can increase the time used for internal message processing by Avaya VTCPD.
	-m a	Specifies asynchronous data delivery. When Avaya VTCPD receives a host data response and an application **RECEIVE** request has not been sent to Avaya VTCPD, Avaya VTCPD sends to the application the condition *unexdata* along with the host data response.
	-m n	Specifies asynchronous data notification. When Avaya VTCPD receives a host data response and an application **RECEIVE** request has not been sent to Avaya VTCPD. However, the application does not accept host data before it sends its **RECEIVE** request. Avaya VTCPD holds the host data (for the application) and delivers the host data after the application sends its **RECEIVE** request.
	-m l	Requires Avaya VTCPD to route the application replies on the same link that the host commands were received on. This option is intended for environments where there is one host, and host commands drive the application. This cannot be used in environments where other host requests can arrive before the application sends its reply to a host command. (See **-m L** below.)
	-m L	Requires Avaya VTCPD to route the application replies on the particular link that is specified by the applications. This is intended for environments where multiple hosts send requests while an application is still processing a prior request. This option instructs Avaya VTCPD to prepend each message with 12 ASCII bytes in the format XXX:YYY:ZZZ, which denotes the connection number, slot number, and Avaya VTCPD port number. The application reply to Avaya VTCPD contains a prepended header that instructs Avaya VTCPD to deliver the reply on a particular link.
	-m I	Specifies that when an application executes a **GET** request, the slot is not reserved. Slots are allocated dynamically in this mode.
	-m R	Specifies that when an application executes a **GET** request, the slot is reserved until it is freed. Slots are allocated on a round robin basis.

Monitoring Host Connections

In case a host process fails, the kernel on the host machine notifies Avaya VTCPD, which sends a **hostdown** condition to applications. However, Avaya VTCPD is not notified if the entire host machine crashes or if the LAN connection is broken. To detect this situation, Avaya VTCPD must query the host periodically with a **ping** command. If a reply is not received within a specified amount of time, Avaya VTCPD determines that the host is down.

Monitoring Host Connections

vtcpd	**-T <timeout>[:<ping-len>] -h len:<str>**	
Args:	**-T <timeout> [:<ping-len>]**	Specifies that a **ping** message is to be sent to the host every **ping-len** seconds. (The default is 3.) If a reply is not received within **timeout** number of seconds, the host is deemed to be down.
	-h len:<str>	Specifies the format of the ping message. For message formats using delimiters or length headers (see *Message Format* on page 46) the length of the ping message can be equal to zero (in which case, only the header or delimiter is sent to the host. The **str** option is ignored). For fixed-length format, the length of the ping message cannot be zero, but can be smaller than the length of the message specified by the **-f** option. If **str** is not specified, the ping message is filled with ASCII '0' (e.g., **-h4**, **-h4:0** and **-h4:0000** are equivalent). To specify an unprintable character, the hexadecimal representation (**0xXX**) should be used.
Examples:	**vtcpd -T 30:10 -h 4:0x00** Generates a **ping** message every 10 seconds with a total timeout of 30 seconds. The message consists of four binary zeroes.	
	vtcpd -T 20:5 -h 4:A0x410x42B Generates a **ping** message every 5 seconds with a total timeout of 20 seconds. The message is the string **"AABB"**.	

Backup LAN

To ensure continued host connectivity, Avaya VTCPD supports the use of a backup LAN between the node(s) and host machines. Avaya VTCPD switches to the backup LAN if host connections through the primary LAN are lost. For multiple hosts and port numbers, repeat the option for each host/port number.

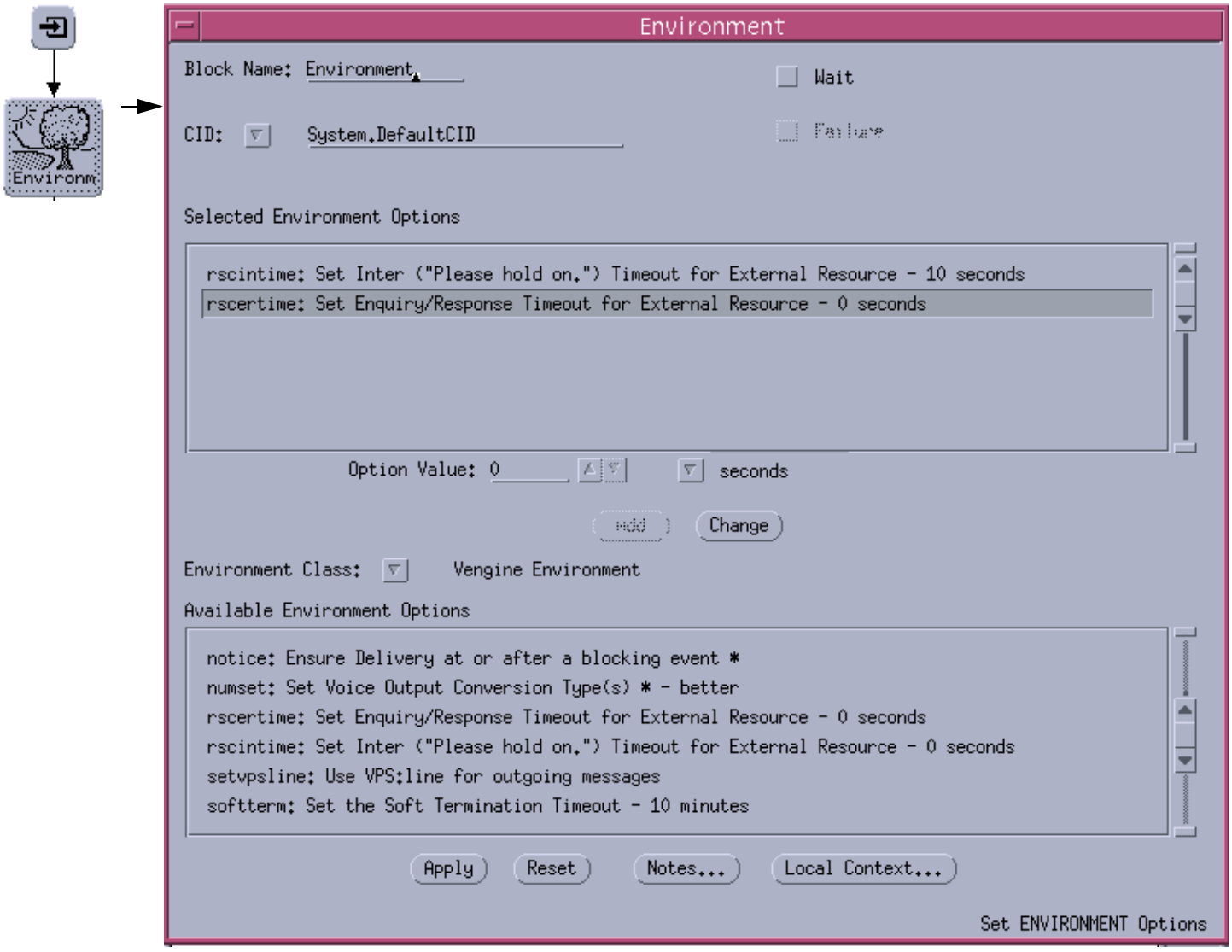
vtcpd	**-L [#:]host:port**	
Args:	**#**	Indicates the number of connections to the specified host. (The default is **1**.)
	host	Indicates the name of each host as it is known on the secondary network.
	port	Indicates the TCP/IP port number. Port numbers 7000-7256 are reserved for the VMST and must not be used.

Time-Outs

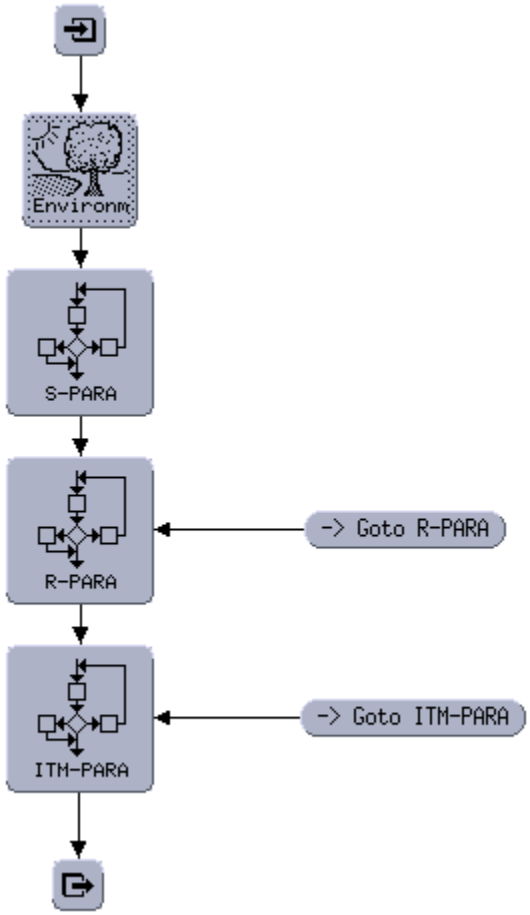
The following VENGINE command line option specifies timeout values in seconds:

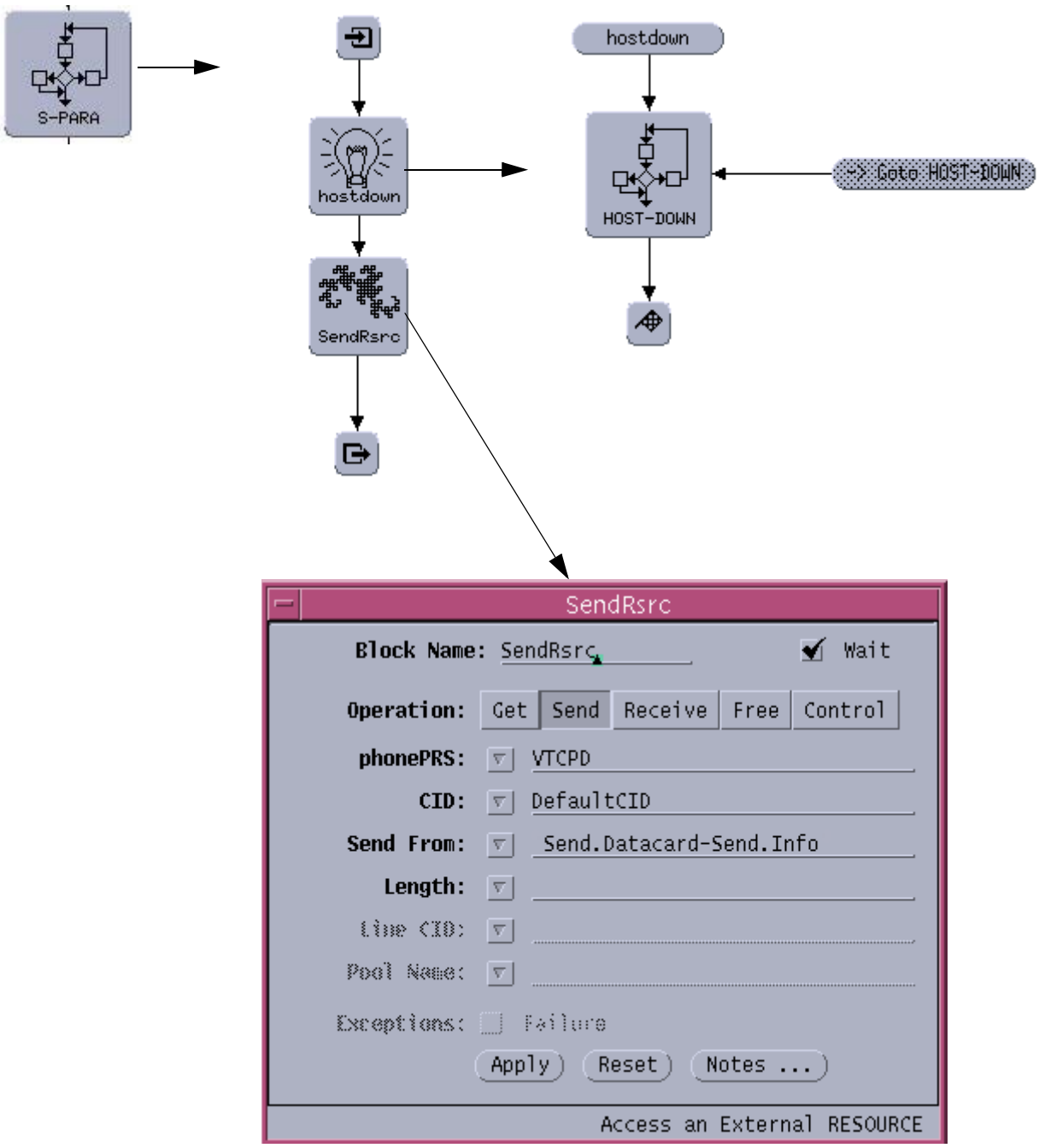
-Q intertimeout:ertimeout

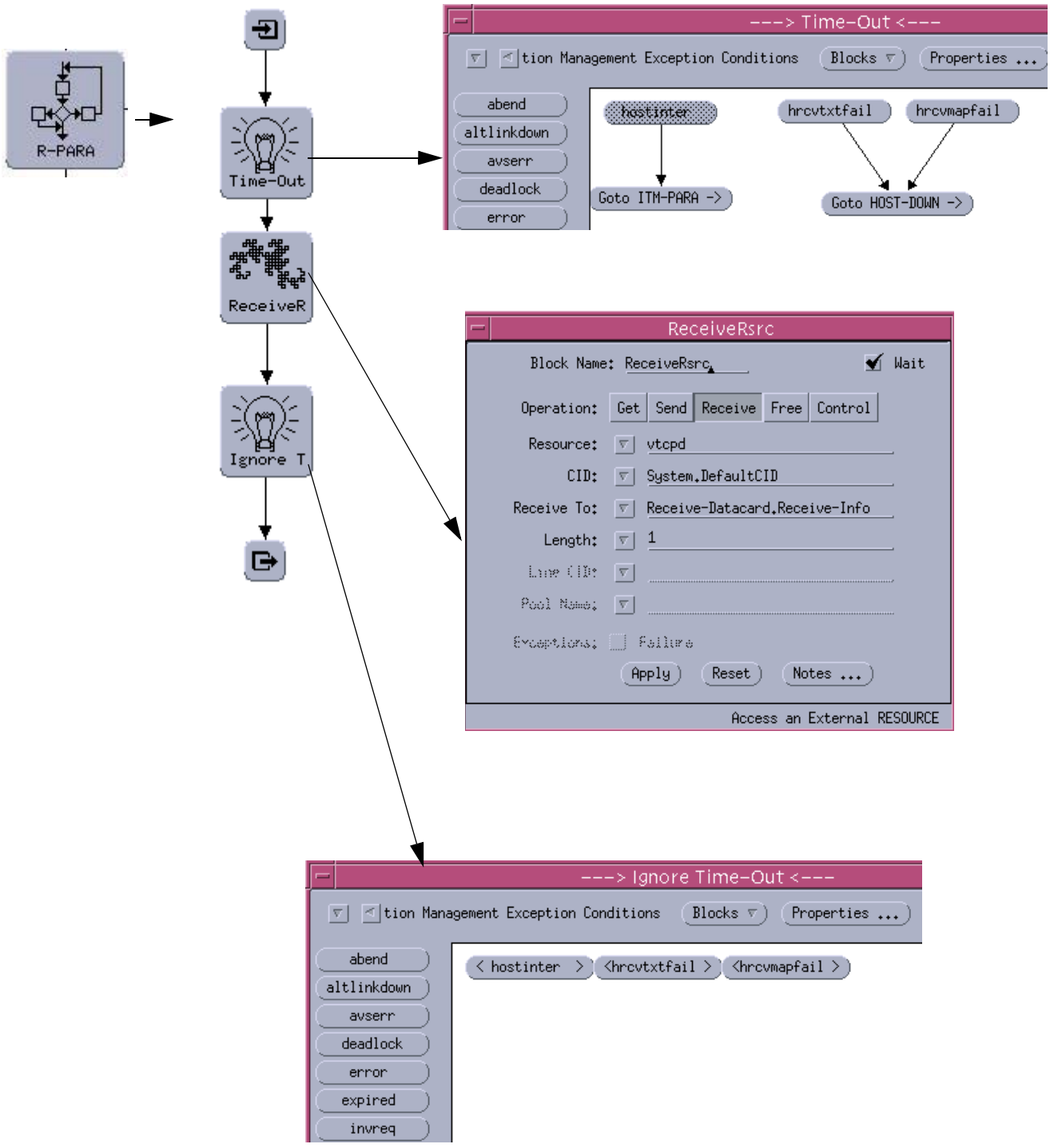
Specify **0** to disable the feature. Time-outs are disabled by default. For example **-Q 10:0** specifies 10 second `intertimeout` and disables **errtimeout**.Time-outs also can be changed dynamically from the application by **ENVIRONMENT** options:

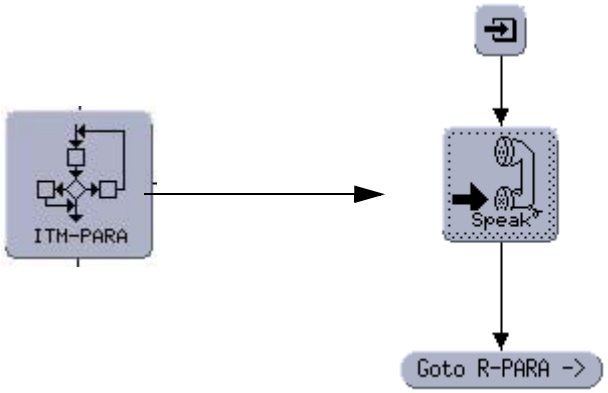

Environm
Environment
Block Name: Environment
Wait
CID: System.DefaultCID
Failure
Selected Environment Options
rscintime: Set Inter ("Please hold on.") Timeout for External Resource - 10 seconds
rscertime: Set Enquiry/Response Timeout for External Resource - 0 seconds
Option Value: 0
seconds
Add
Change
Environment Class: Vengine Environment
Available Environment Options
notice: Ensure Delivery at or after a blocking event *
numset: Set Voice Output Conversion Type(s) * - better
rscertime: Set Enquiry/Response Timeout for External Resource - 0 seconds
rscintime: Set Inter ("Please hold on.") Timeout for External Resource - 0 seconds
setvpsline: Use VPS:line for outgoing messages
softterm: Set the Soft Termination Timeout - 10 minutes
Apply
Reset
Notes...
Local Context...
Set ENVIRONMENT Options

When time-outs are specified, the VENGINE starts its internal timer when a **`RECEIVE RESOURCE`** request is sent. If the intertimeout expires, the VENGINE plays an item defined by the command line option **`-o`** (default 'Please hold on.') and restarts intertimeout. When errtimer expires the VENGINE generates `ertimeout` condition.

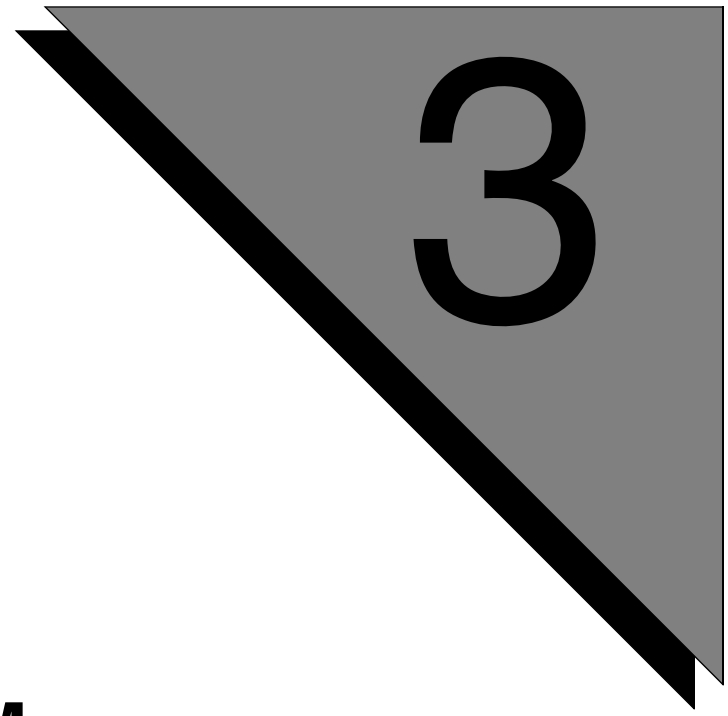

Environm
S-PARA
R-PARA
-> Goto R-PARA
ITM-PARA
-> Goto ITM-PARA


S-PARA
hostdown
SendRsrc
hostdown
HOST-DOWN
-> Goto HOST-DOWN
SendRsrc
Block Name: SendRsrc
Wait
Operation: Get Send Receive Free Control
phonePRS: VTCPD
CID: DefaultCID
Send From: Send.Datacard-Send.Info
Length:
Use CID:
Pool Name:
Exceptions: Failure
Apply Reset Notes ...
Access an External RESOURCE


R-PARA
Time-Out
ReceiveR
Ignore T
---> Time-Out <---
tion Management Exception Conditions
Blocks
Properties ...
abend
altlinkdown
avserr
deadlock
error
hostinter
hrcvtxtfail
hrcvmapfail
Goto ITM-PARA ->
Goto HOST-DOWN ->
ReceiveRsrc
Block Name: ReceiveRsrc
Wait
Operation: Get Send Receive Free Control
Resource: vtcpd
CID: System.DefaultCID
Receive To: Receive-Datacard.Receive-Info
Length: 1
Apply
Reset
Notes ...
Access an External RESOURCE
---> Ignore Time-Out <---
tion Management Exception Conditions
Blocks
Properties ...
abend
altlinkdown
avserr
deadlock
error
expired
invreq
< hostinter >
<hrcvtxtfail >
<hrcvmapfail >



When the application receives the data for **`RECEIVE`** request, the daemon cancels all pending time-outs. Nevertheless, because of possible race conditions, it is still possible that a condition can be sent after the data. To avoid confusion, it is advisable that the application ignores `intertimeout` and `ertimeout` conditions after receiving the data.



VTCPD Messages

This chapter covers:

1. **Message Format**
2. **Message Routing**
3. **ISSUE-SEND-RECEIVE (ISR)**
4. **ISSUE-RECEIVE-SEND (IRS)**

Message Format

Two message formats are available - fixed length, and variable length. Fixed length messages have a predefined length, and most variable length messages use a special character (or a string) to specify the end of the message. However, the variable length message with a fixed byte message header message format uses a predefined header to specify message length.

Message Lengths

Format	Description
fixed length	The message contains a fixed number of bytes.
variable length	Variable length message delimited using a special character or a string.
	Variable length message with a fixed byte message header that defines the message length.

Message formats can be specified by:

- **`-f l:lenfld [:pos [:hdrlen] ]`** option for binary length fields.
- **`-f a:lenfld [:pos [:hdrlen] ]`** option for ASCII length fields.

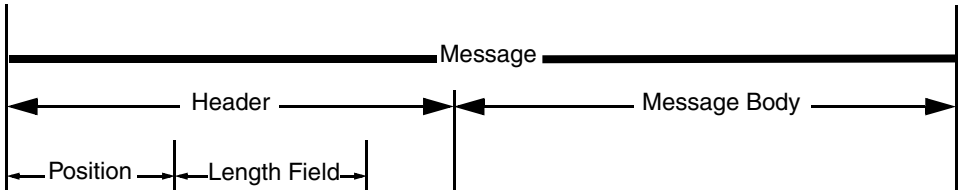
The following is important information regarding the use of these options:

- If both fixed-length and variable-length messages must be processed, then two instances of the daemon must be used, each started with a different value for the **`-n <name>`** parameter.
- VTCPD adds a message header or a delimiter to a string received from an application. The application should not include a message header or a delimiter with the message sent to the daemon.

The length field contains the length of the message body. However, if options **`-f L`** or **`-f A`** are specified instead of **`-f l`** or **`-f a`**, the length field contains the total message length (message header and body). The position is counted from 1 (default); **`hdrlen`** defaults to **`lenfld`** + **`pos`** - 1.

The position of the message length information is predefined in the header and the system goes to the specified Length Field to obtain the message length specification. For example, the Position of the message Length Field can be defined as beginning on the fourth character of the header and the Length Field can be defined as being six characters long. The system proceeds to the fourth character, and reads the next six characters to obtain the message length.

Message Format



The default setting does not pass the header to the application. VTCPD strips the header before passing the message, and prepends the header before sending to the host. The command line option **`-m h`** is specified to instruct VTCPD not to separate the header from the body, and to pass the header to the application. For example:

- **`-f a:4:1:4`** is equivalent to **`-f l:4`** and means that the message header contains only the length field of 4 bytes in ASCII format, and the header is stripped when passed to the application.

- **`-f l:2:1:8 -m h`** means that the message header contains 8 bytes, the binary length field of 2 bytes is located at the beginning of the header, and the header is passed to the application.

- If the message body contains 10 bytes and the header has 8 bytes, then depending on the options, the contents of the **`lenfld`** and the number of bytes passed to/from the application are as follows:

`-f` Option	**Action**	**strip header**	**retain header (`-m h`)**
`{l|a}:lf`	send to host	18 bytes	18 bytes
	send to application	10 bytes	18 bytes
	value of `lf`	10	10
`{L|A}:lf`	send to host	18 bytes	18 bytes
	send to application	10 bytes	18 bytes
	value of `lf`	18	18

Message Format Options

```
vtcpd    -f {# | d:D | l:{#|lf[:hp[:hl]]} |
                       L:{w|lf[:hp[:hl]]} |
                       a:{n|lf[:hp[:hl]]} |
                       A:{n|lf[:hp[:hl]]}
          [{-m h] | -m t}]
```

Args:	**#**	Specifies a fixed-length format with the indicated number of bytes.
	d:D	Specifies the delimiter **D**, which can be a printable character or **0xXX** format to specify the character's a hexadecimal number.
	-m h	Specifies that VTCPD is to retain the header in the message.
	-m t	Specifies that VTCPD is to truncate all trailing spaces from the string prior to sending it to the host. This option should not be used with fixed-length messages.
	l:#	Specifies the length (in binary format) defined in bytes that precede the message (1, 2, or 4). Note that for inter-platform compatibility, the length must be specified in the order recognized by the network (big endian or most significant byte).
	l:lf[:hp[:hl]]	Specifies that the (binary) length field is embedded within the message header. The length field (**lf**) contains the length of the body portion of the message. The header position (**hp**) is counted starting at 1 (which is the default). The header length (**hl**) defaults to **lf**+**hp**-1. By default, VTCPD strips the header before passing the message to the application, and prepends it before sending to the host.
	L:Lf[:hp[:hl]]	Specifies that the (binary) length field is embedded within the message header. The length field (**Lf**) contains the total message length (which is the header plus body portion). The header position (**hp**) is counted starting at 1 (which is the default). By default, VTCPD strips the header before passing the message to the application, and prepends it before sending to the host.
	a:lf[:hp[:hl]]	Specifies that the (ASCII) length field is embedded within the message header.

Message Format Options

vtcpd **-f {# | d:D | l:{#|lf[:hp[:hl]]} | L:{w|lf[:hp[:hl]]} | a:{n|lf[:hp[:hl]]} | A:{n|lf[:hp[:hl]]} [{-m h] | -m t}]**

Args (cont):	**A:Lf[:hp[:hl]]**	Specifies that the (ASCII) length field is embedded within the message header.
	A:n	Specifies the length (in ASCII format) of the message header, which contains the data length in ASCII format.
	a:n	Specifies that the length in the header field includes the length of the data in addition to the length of the header.
Examples:	**vtcpd -f 120** The message has a fixed length of 120.	
	vtcpd -f d:0x0A The messages are delimited by newline characters (0x0A).	
	vtcpd -f l:2 -m t The message length is contained in a two-byte header. Any trailing spaces in the message are removed before it is sent to the host.	
	vtcpd -f A:3 The message length is contained in a three-character header. For example, if the message sent by the host is "ABCDE", the message as received by an application is "008ABCDE".	
	vtcpd -f a:3 The combined message and data length is contained in a three-character header. For example, if the message sent by the host is "ABCDE", the message as received by an application is "005ABCDE".	
	vtcpd -f a:4:1:4 *-or-* **vtcpd -f l:4** The message header contains only the length field (4 bytes in ASCII format) and the header is stripped when passed to the application.	
	vtcpd -f l:2:1:8 -m h The message header contains 8 bytes, the binary length field of 2 bytes is at the beginning of the header, and the header is passed to the application.	

Message Identification (ID)

VTCPD uses a Message ID to associate received messages with application requests. Requests sent to the host from an application and the response to the application are identified as corresponding to each other because they use the same Message ID. Message IDs are not used when there is only one connection per line, because in this configuration, the line identifies the source and destination of the message.

Messages are identified by unique combination of N consecutive bytes, starting from the M -byte position of the message. This is called the Message ID. Unless a host connection is exclusively reserved for a phone line (see below), all messages must include message ID.

VTCPD uses one of the following methods to assign the Message ID:

Message ID

Bytes	Message ID	Example
one	Binary phone line number of the application sending the request	**`-I 2`**
two	Binary phone line number and MPS number of the requesting application.	**`-I 2:1`**
three	ASCII phone line number of the requesting application in the format 999.	**`-I 201`**
six	ASCII phone line and MPS number of the requesting application.	**`-I 6:24`**

If the Message ID is assigned by applications, they are responsible for assuring that no two outstanding requests have the same ID. The daemon doesn't check the context of the ID field. It only matches it byte for byte with the ID in host responses. If two requests with the same ID are pending at the same time, the results will be unpredictable.

If host data contains a Message ID that cannot be associated with any phone line, the data is discarded. However, it is possible to pre-assign an administrative phone line that receives all unidentified messages. The messages are sent to the line as the data portion of the condition **`unexdata`**.

Message ID Options

vtcpd	**{-i|I} {<len>:<pos>|<len>@<str>} [-m i] [{-a <#>|-A <#>}]**	
Args:	**-i**	The Message ID is assigned by applications.
	-I	The Message ID is assigned by VTCPD.
	<len>	Specifies the length in bytes for the message ID field.
	<pos>	Specifies the starting position of the ID field (i.e., the offset).
	<str>	Specifies a string within the message that identifies the Message ID field by directly preceding it. The string can be any set of ASCII characters except 0x00. Unprintable characters are represented in hexadecimal as 0xXX.
	-a <#>	Specifies the line number of an administrative application that receives any unidentifiable messages.
	-A <#>	Specifies the line number of an administrative application that sends all messages from the host. In this case, the Message ID fields from application GET and SEND requests are ignored.
	-m i	Specifies that VTCPD is to run in *inverted* mode. This is intended for messages that cannot be correlated by means of the Message ID field. Use this option if the message IDs are set by the host.
Examples:	**vtcpd -i 48:1** The Message ID is supplied by the application. The Message ID field is 48 bytes in length and starts from the beginning of the message.	
	vtcpd -I 2:1 The Message ID is supplied by VTCPD and consists of binary representations of the phone line and MPS numbers.	
	vtcpd -I 6:24 The Message ID is supplied by VTCPD. The Message ID consists of ASCII representations of the phone line and MPS numbers.	
	vtcpd -i 4@abc0x0A0x0B The Message ID is supplied by the application. The Message ID field is four bytes in length, and is located directly after the string `abcXY`, where X and Y are binary values representing 1 and 2.	

All messages must include a Message ID field. The length and location of the ID field is specified by:

```
-i N:M
```

option:

For example, -i 48:1 specifies an identification field of length 48, starting from the beginning of the message. Alternatively, identification fields for requests can be assigned by applications (using -i) or by the daemon (by using -I instead).

Instead of an offset, the location of the ID field in the message can also be specified by a identification string which precedes the ID. To define this option use a command line argument:

`-i len@string` or **`-I len@string`**

instead of `-i len:pos` or `-I len:pos`. The string can be any set of ASCII characters except 0x00. Unprintable characters may be represented in hexadecimal as 0xXX. For example, -i 4@abc0x010x02 specifies that the ID field in a message is located right after a string "abcXY", where in this example X and Y are binary values of 1 and 2 respectively.

Format of Outgoing and Incoming Messages

Typically, the header format is the same for both outgoing and incoming messages. If formats differ, then options **`-f`** and **`-i`** are applied to outgoing messages only, while new options **`-F`** and **`-j`** should be used to specify the format of incoming messages. If either of **`-F`** or **`-j`** options is omitted, the format is defined by corresponding option **`-f`** or **`-i`**.

Message Routing

Connection capacity means the number of outstanding host requests that can be associated with the connection. The following choices are supported:

Connection Capacity

Capacity	Description
Unlimited	Any number of outstanding host requests can be associated with the link.
Limited	Each connection has a pool of slots to be granted to phone lines by the daemon.
Common Pool	All host links have a common pool of slots, which means that the total number of outstanding host requests is limited.

Currently, only common pool or per link pool can be specified. Since the number of phone lines per MPS is up to 999, there cannot be more than 999 outstanding requests. Consequently, unlimited capacity links are treated as links with pool size 999.

The capacity is specified by daemon's command line options `-p` (for per line pool) or `-P` (for common pool). Only one of these options is allowed. If neither of them is specified or if the value is equal to or greater than 999, the connections will have unlimited capacity (the default).

Message Routing Options

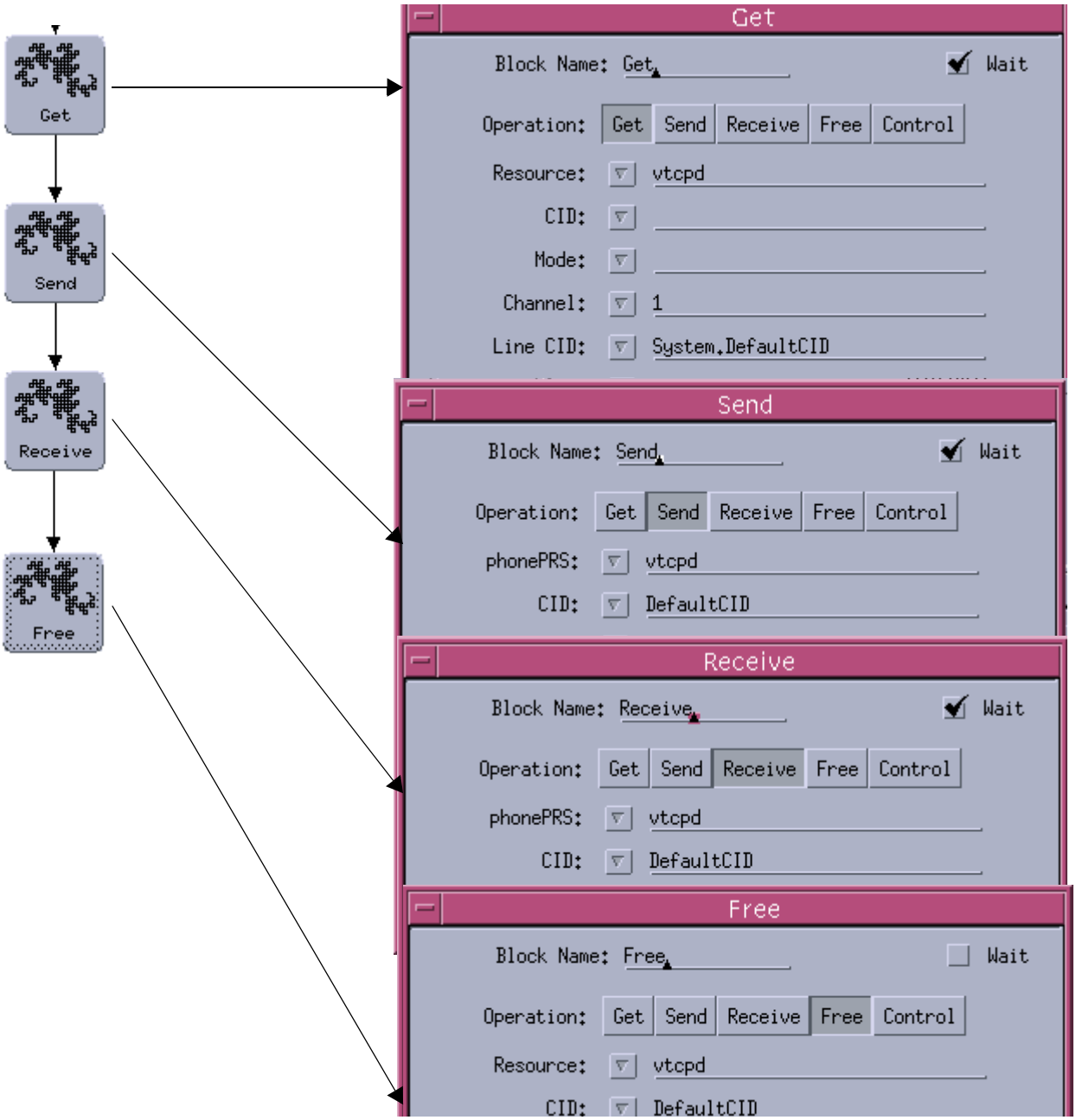
vtcpd	**[{-p <#>|-P <#>}] [-m R]**	
Args:	**-p**	Lowercase "**p**" specifies limited capacity, where each connection has a pool of slots.
	-P	Uppercase "**P**" specifies a common pool of slots for all connections.
	-m R	Specifies that VTCPD chooses the connection and the corresponding slot in the pool using round robin load balancing (instead of the default method of round robin balancing among slots with the minimum number of outstanding requests.) If all slots are used the request is rejected.
Examples:	**vtcpd -l 20:vpshost:1000 -P 20** Specifies a global pool of 20. It is possible that all 20 requests may be sent through one connection, leaving all other links idle.	
	vtcpd -l 20:vpshost:1000 -p 1 Specifies one request per connection. Messages may be sent without IDs.	

ISSUE-SEND-RECEIVE (ISR)

The **ISSUE-SEND-RECEIVE** model assumes the ideal situation, that all host data is solicited or that the application knows (or assigns) the Message ID field.

This may be not the case for host driven applications when the host assigns Message ID. In these cases, the host may need to execute on any available phone line so the **ISSUE-RECEIVE-SEND** model has to be used.

A typical application **ISSUE-SEND-RECEIVE** scenario includes:



This allows the daemon to:

- Reserve an available host connection.
- Send data to the host and associate Message ID with the phone line (**SEND**)

A slot (receiving a **GET** request) receives data from the host based on the saved Message ID (**RECEIVE**) and releases the slot (**FREE**).

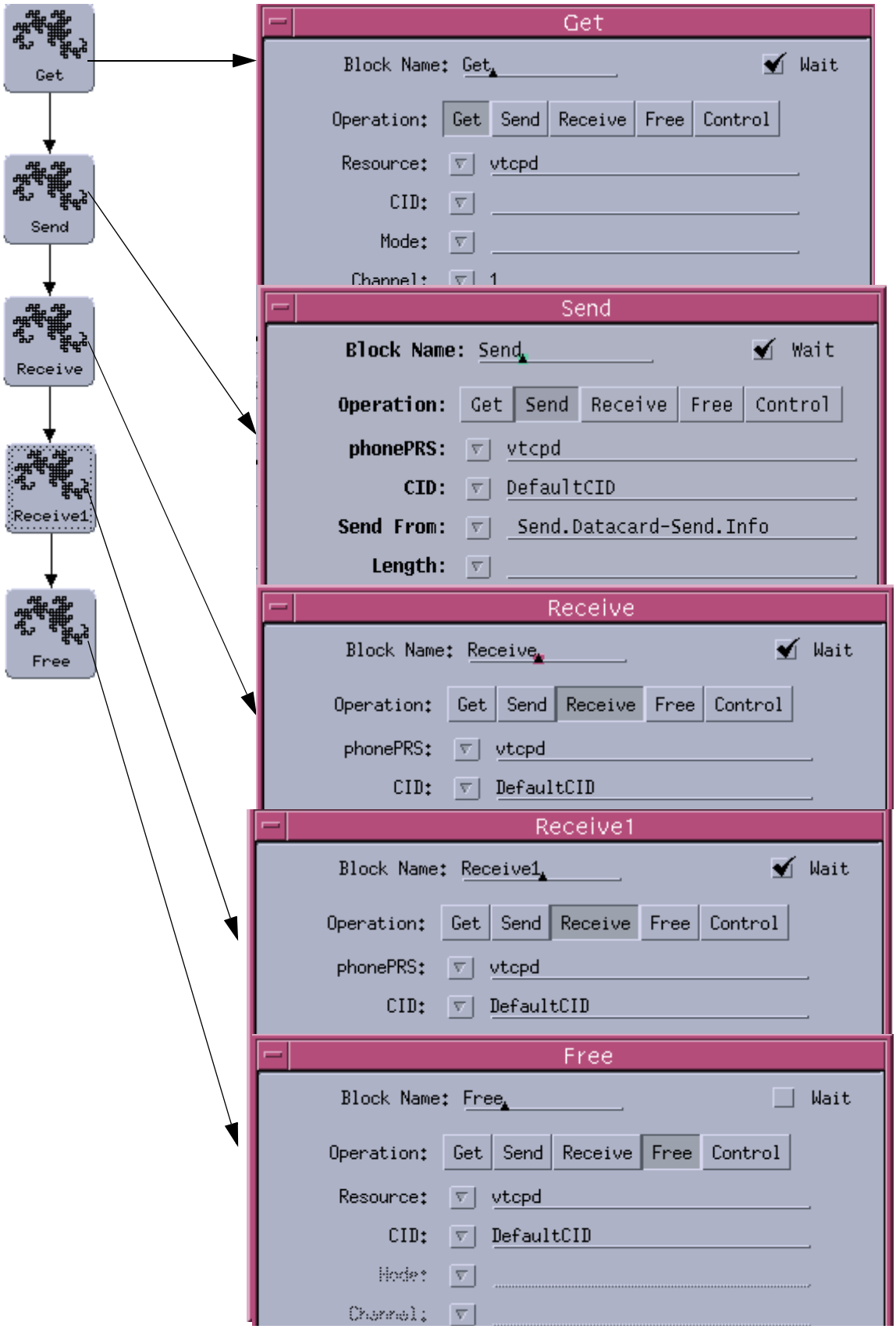
As a reply to **GET**, the data portion of the condition **gotres** contains the connection number, slot number and VTCPD daemon's port number in the format XXX:YYY:ZZZ. (The application doesn't need to process the data.)

If a reply from the host arrives before the application issues **RECEIVE**, the reply is held by the VTCPD daemon.

If a line executes termination or recycling without **FREE**, the slot is released automatically.

In any message format, the **SEND ()** string should include only data, whereas the delimiter (**-f d:** format) or a length header (**-f l:** format) is provided by the VTCPD daemon.

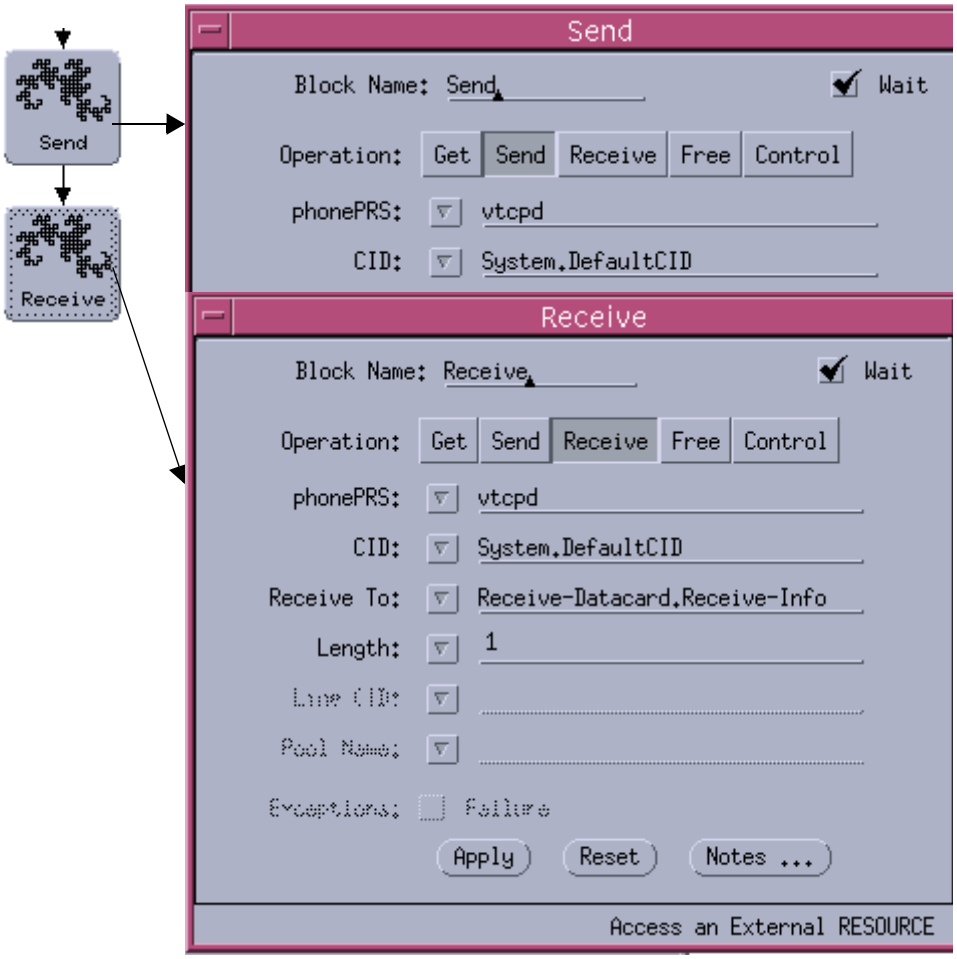
After the initial **SEND**, it is possible to mix **SEND** and **RECEIVE** operations in any order. The Message ID of the last **SEND** is always in effect (until **FREE**) and can be used to identify consecutive **RECEIVE**s. For example:


Get
Send
Receive
Receive1
Free
Get
Block Name: Get
Wait
Operation: Get Send Receive Free Control
Resource: vtcpd
CID:
Mode:
Channel: 1
Send
Block Name: Send
Wait
Operation: Get Send Receive Free Control
phonePRS: vtcpd
CID: DefaultCID
Send From: Send.Datacard-Send.Info
Length:
Receive
Block Name: Receive
Wait
Operation: Get Send Receive Free Control
phonePRS: vtcpd
CID: DefaultCID
Receive1
Block Name: Receive1
Wait
Operation: Get Send Receive Free Control
phonePRS: vtcpd
CID: DefaultCID
Free
Block Name: Free
Wait
Operation: Get Send Receive Free Control
Resource: vtcpd
CID: DefaultCID
Mode:
Channel:

This allows the line to receive two messages with the specified Message ID-field.

Reserving a Slot

Slots can be required to identify limitations on connectivity. VTCPD has unlimited connectivity. However, if a host can only receive five simultaneous messages on a line then there are five slots defined for that line. VTCPD identifies that line as having only five slots available and reserves slots on the line as necessary to insure that the messages it needs to send can be sent. The slot is released when the host data arrives.



The command option **`-m I`** is used when a permanent slot allocation is not desired. It allows the application to use the **Get/ Free** operation of the PeriProducer resource block without the associated slot allocation.

Queuing Requests

The VTCPD daemon has two options when it receives a request from an application. It can send the request to the host (default), or keep the request locally until the previous request(s) is answered.

If the message is sent to the host, requests are either physically stored on the host side or in the kernel buffer. If an application restarts before receiving the reply, this has no effect on the host, which eventually retrieves and processes the request. However, this could result in the processing of outdated requests which can further reduce already limited host resources.

VTCPD can also hold requests locally and remove them from the queue when the application restarts. This is done if the host is unable to process several messages simultaneously. The VTCPD command line option is **`vtcpd -q #`**

This option specifies how many unanswered requests are sent to the host before queuing them locally. If the option is not specified, all requests (default) are immediately sent to the host. When **`-q`** option is specified, the VTCPD either sends the request or stores it in a queue. When a reply comes from the host, the VTCPD forwards the first message from the queue to the host. Another option used to maintain one global queue for all links is **`vtcpd -Q #`**

In the global queue option, all requests go into the single queue and are retrieved when the number of unanswered requests for any link becomes smaller than the number specified by **`-Q`** option. In this mode, any request can be sent through any link, so the **`-m I`** option is added to the list of command line arguments and the option **`-m R`** (if any) is disabled.

In the global mode, any request can be sent through any link, so the **`-m I`** option is added to the list of command line arguments. Correspondingly, the option **`-m R`**, (if specified) is disabled.

Queueing Requests

`vtcpd`	`[{-q <#> | -Q <#>}]`	
Args:	`-q <#>`	Specifies how many unanswered requests go to the host before they are queued locally. Unless this option is specified, all requests are immediately written to the host.
	`-Q <#>`	Specifies that all requests from all links are to be placed in a global queue. When the number of unanswered requests for any link becomes smaller than the specified value, the requests are retrieved.

If an application restarts and executes either the **Free** or **Get** operation, all related requests are removed from the queues.

Clearing the Connection

When VTCPD queues requests, it does not send them to the host until the previous request has been answered. However, requests can go unanswered due to a variety of reasons (for example, incorrectly formatted requests, bugs in the code) The unanswered request causes the corresponding connection to be unusable.

The option **-d sec** allows VTCPD to start using the connection again after a specified time period. The option **-D sec** is used to close the specified connection. This permits the connection to be reused.

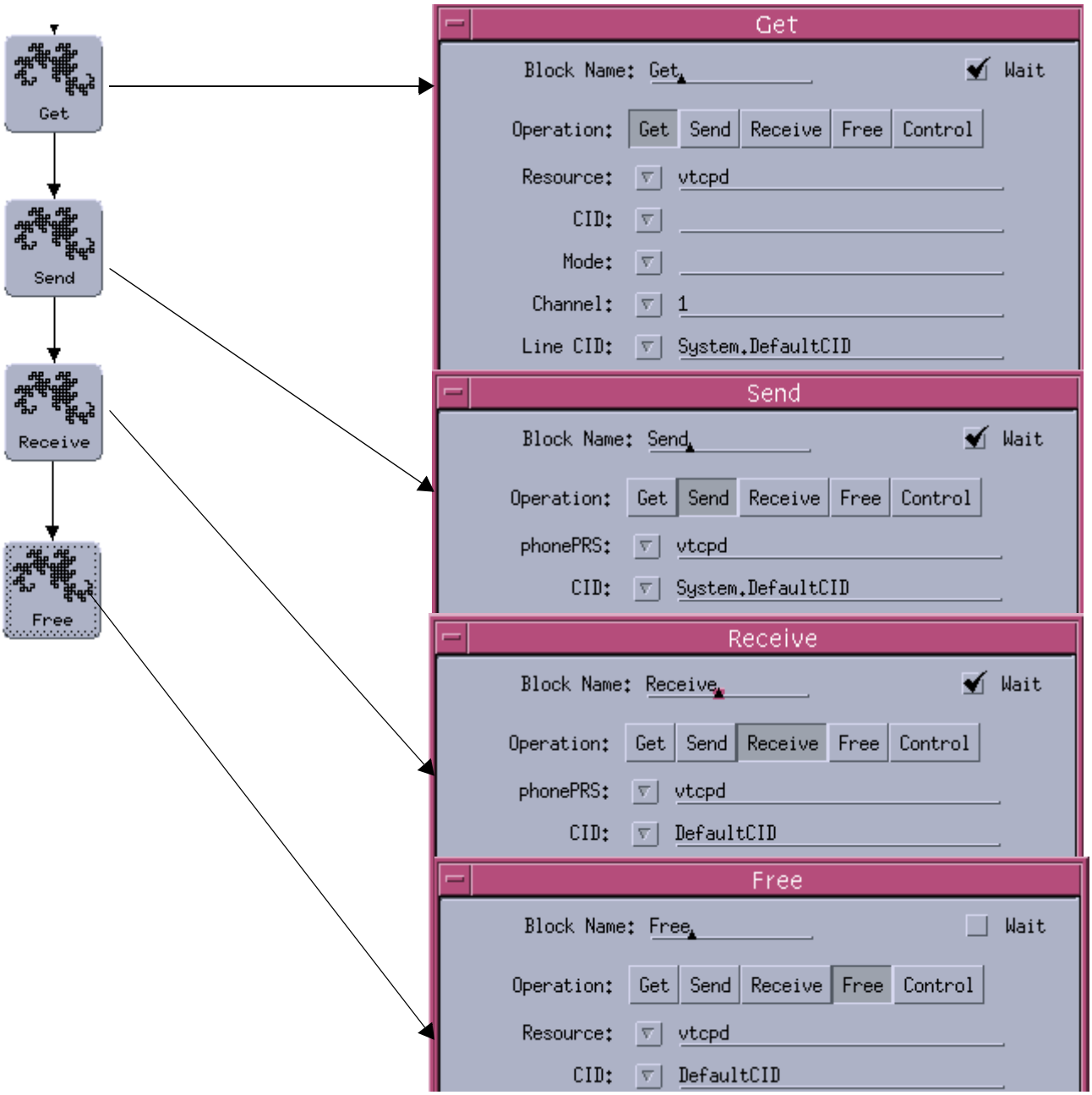
Changing Data Length

The optional **LENGTH** parameter in the application **SEND** command allows the number of characters sending to the VTCPD daemon to be specified. This is used when the amount of data is changed and the program is using the same **SEND** buffer. Do not use this parameter with the fixed length message format (option **-f len**). The length specified must not be less then the total length of the message.

The length may also be modified by truncating the trailing spaces using the VTCPD **-m t** option. This option is ignored if the **LENGTH** is specified.

Application's Connection Choice

The application can specify the host connection using:



The channel number # has to be in the range from 1 to the total number of host connections, specified by the daemon's command line option, **`-l`**.

Replies Routing

Typically, a reply arrives from the host through the same connection used to send the request. To specify that the reply can come from any connection, the command line option **`-m d`** is used. Use of this option may cause slightly longer internal message processing by the VTCPD daemon.

Asynchronous Replies and Reply Notification

When a response is received from the host, the VTCPD may handle the response in one of three different ways:

- The response may be immediately delivered to the application.
- The response may be held by VTCPD, until VTCPD receives a request from the application for the response.
- VTCPD can send a message to the application notifying it that the host response is available.

The host response is delivered immediately, if VTCPD already has the **RECEIVE** request from the application or if no **RECEIVE** request is required by the application.

If a **RECEIVE** request from the application is required and it has not been received, then VTCPD can wait for the **RECEIVE** request, or it can send out a response available message to the application.

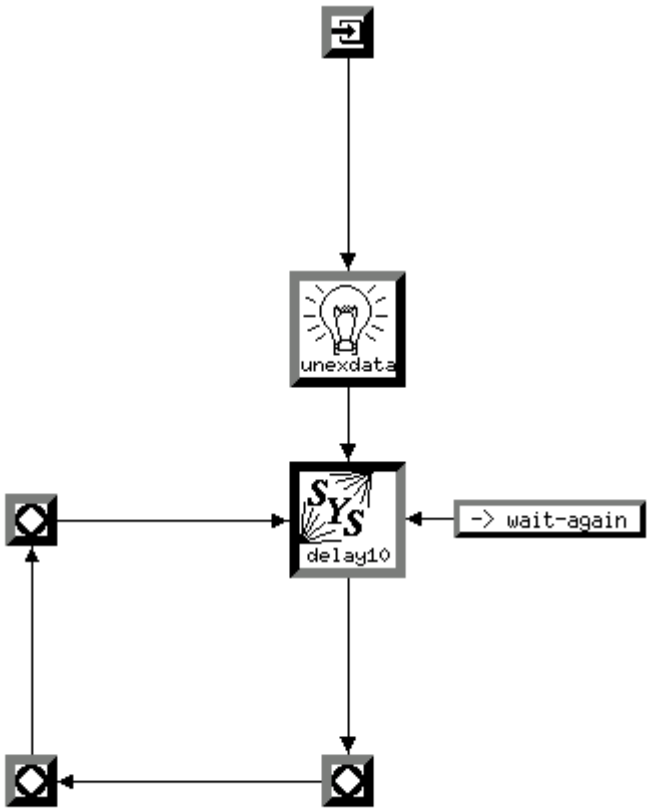
The VTCPD daemon uses asynchronous data delivery to support these conditions. Starting the VTCPD daemon with the command line option **-m a** instructs the VTCPD daemon to send to the application the **unexdata** condition (contains the host response data). Using the **-m n** option changes the condition, now the condition only notifies the application that it can retrieve host response data by issuing a **RECEIVE** operation.

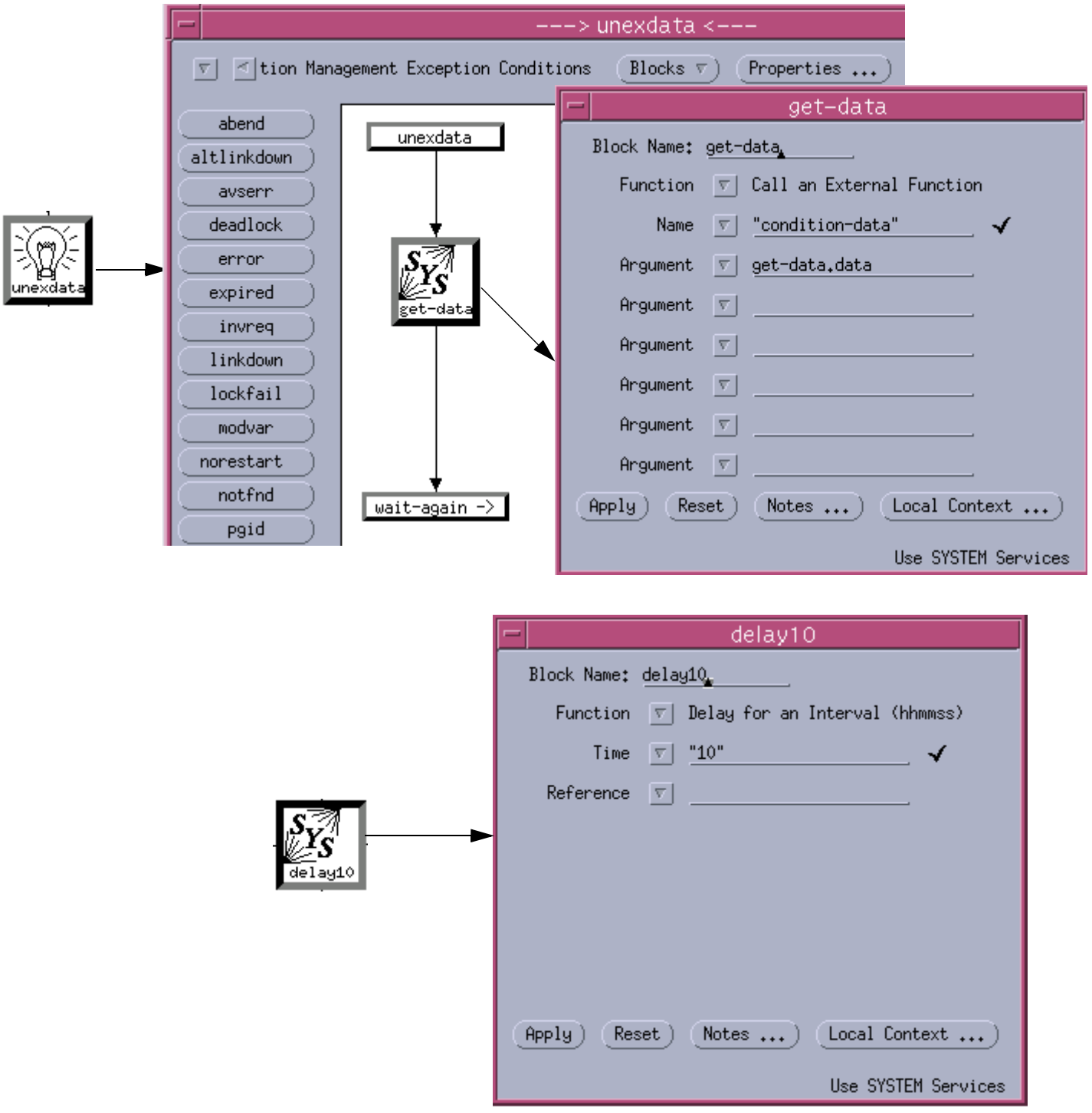
Unidentified Host Data

If host data contains a Message ID which cannot be associated with any phone line, the data is discarded. However, it is possible to pre-assign an administrative phone line which will receive all unidentified messages. This can be specified by the VTCPD command line option **`-a #`**. The messages are sent to the line as the data portion of the condition **`unexdata`**.

These conditions are sent asynchronously to avoid possible race conditions so blocking requests (**`GET, SEND...WAIT`** etc.) must be avoided.

The sample PeriProducer application demonstrates the administrative application:





For administrative applications, start the VENGINE using `option -l 0:0`.

If the line restarts, all messages arriving while the line is in transition will be lost.

Administrative Line

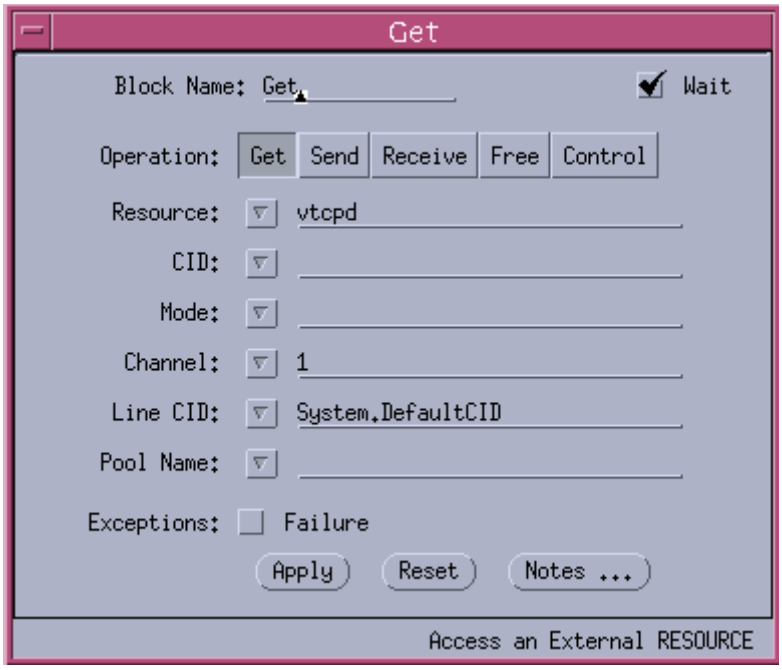
The option **-A #** sends all host data to the specified administrative phone line. The Message ID fields of **GET, CONTROL, FREE** and **SEND** messages from applications are ignored.

ISSUE-RECEIVE-SEND (IRS)

This model implies that all host data is solicited, and the application somehow knows (or assigns) the ID field.

Registering the ID

If the application has the Message ID but has to wait for unsolicited host data, it registers the Message ID using:

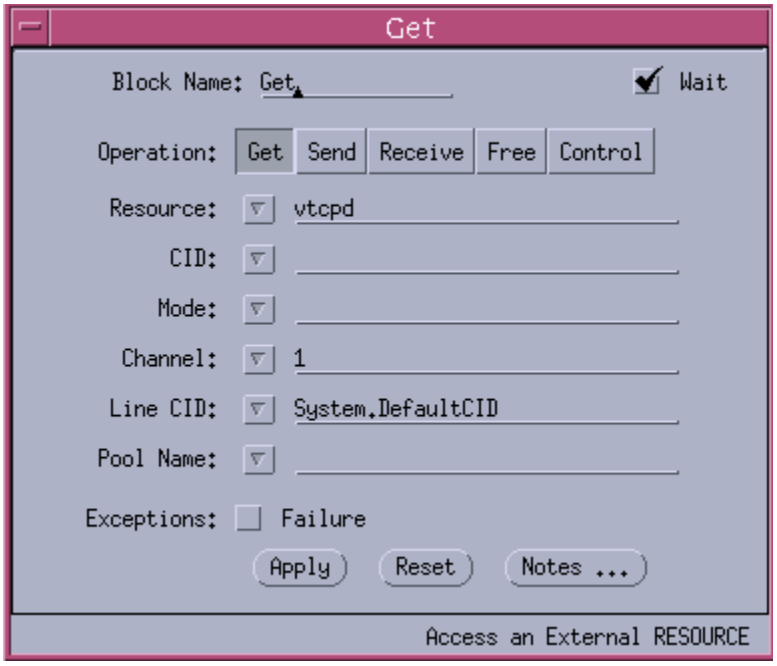


It blocks on **RECEIVE** operation without invoking **SEND**.

Inverted Mode

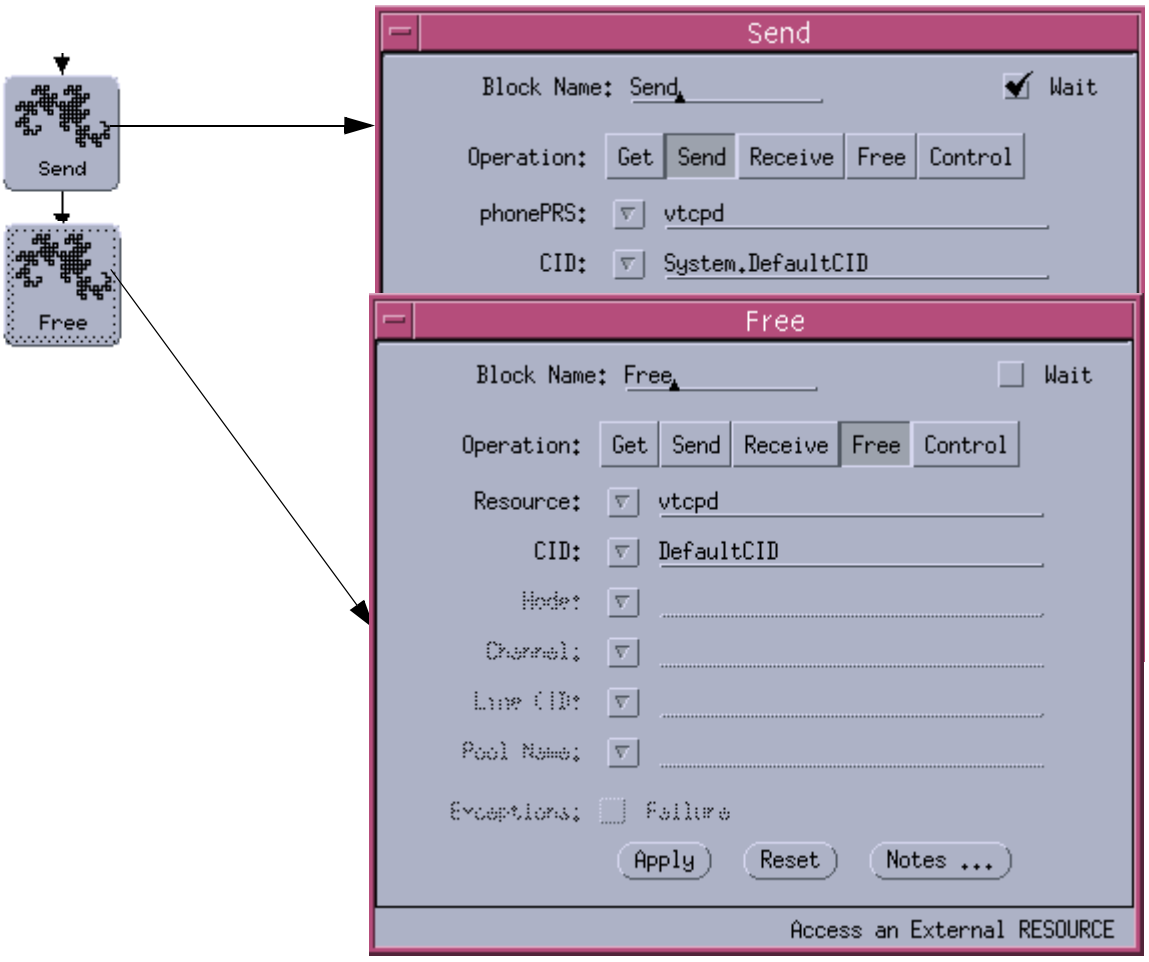
If the application doesn't have the Message ID it informs the VPCPD daemon that it is ready to process the next available host data (which cannot be correlated with any phone line). To support this, the VTCPD daemon runs in the inverted mode, specified by the command line option **-m i**.

When the daemon is running in the inverted mode the application issues a:



This informs the VTCPD daemon of the application's availability. When the host data arrives, the VTCPD daemon tries to locate a phone line with the matching Message ID. If no such line exists, the Message ID is associated with the registered phone line and the data is sent to the application.

The application then executes **SEND** operations which do not change Message ID. All other messages from the host with the same Message ID are delivered to the same line. The line can disassociate itself from the Message ID by executing the **Free** operation of the PeriProducer Resource block.



Application Connection Choice

The VTCPD option **`-m 1`** forces the daemon to use the same link for **`SEND`** routing. This specific link use for **`SEND`** routing may be required if there are specific host processing options, or host process memory requirements that are associated directly with an individual line. This option only works if the host receives no other requests before the application sends its reply.

If several hosts send requests while the application is still processing, the routing information must be included with data sent by the application. To include the routing data, use the VTCPD option **`-m L`**. VTCPD includes the routing data by prepending each message with 12 ASCII bytes in the format XXX:YYY:ZZZ, where XXX is the link number to use (YYY and ZZZ are reserved for future use). The application also prepends this header to all messages sent to VTCPD to guarantee delivery of all messages to the host through the link XXX.

Choosing the Daemon

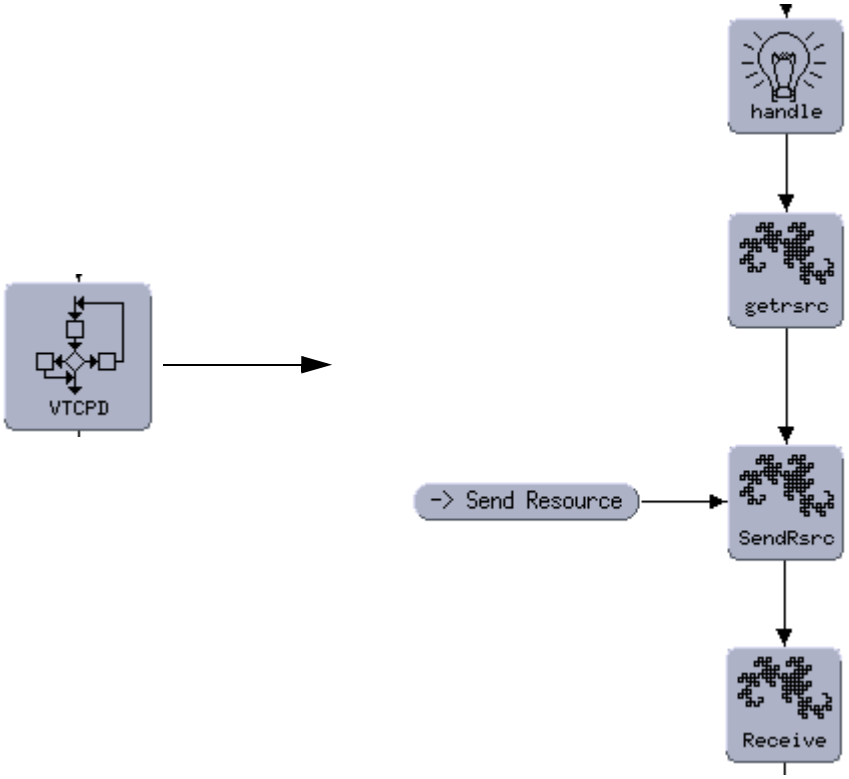
The operations **GET, CONTROL, FREE** or **SEND** broadcast send requests to all VTCPD daemons attached to the vmst. These operations cause no problems so long as only one VTCPD daemon is running. However, problems can arise when more then one VTCPD daemon is attached to the vmst.

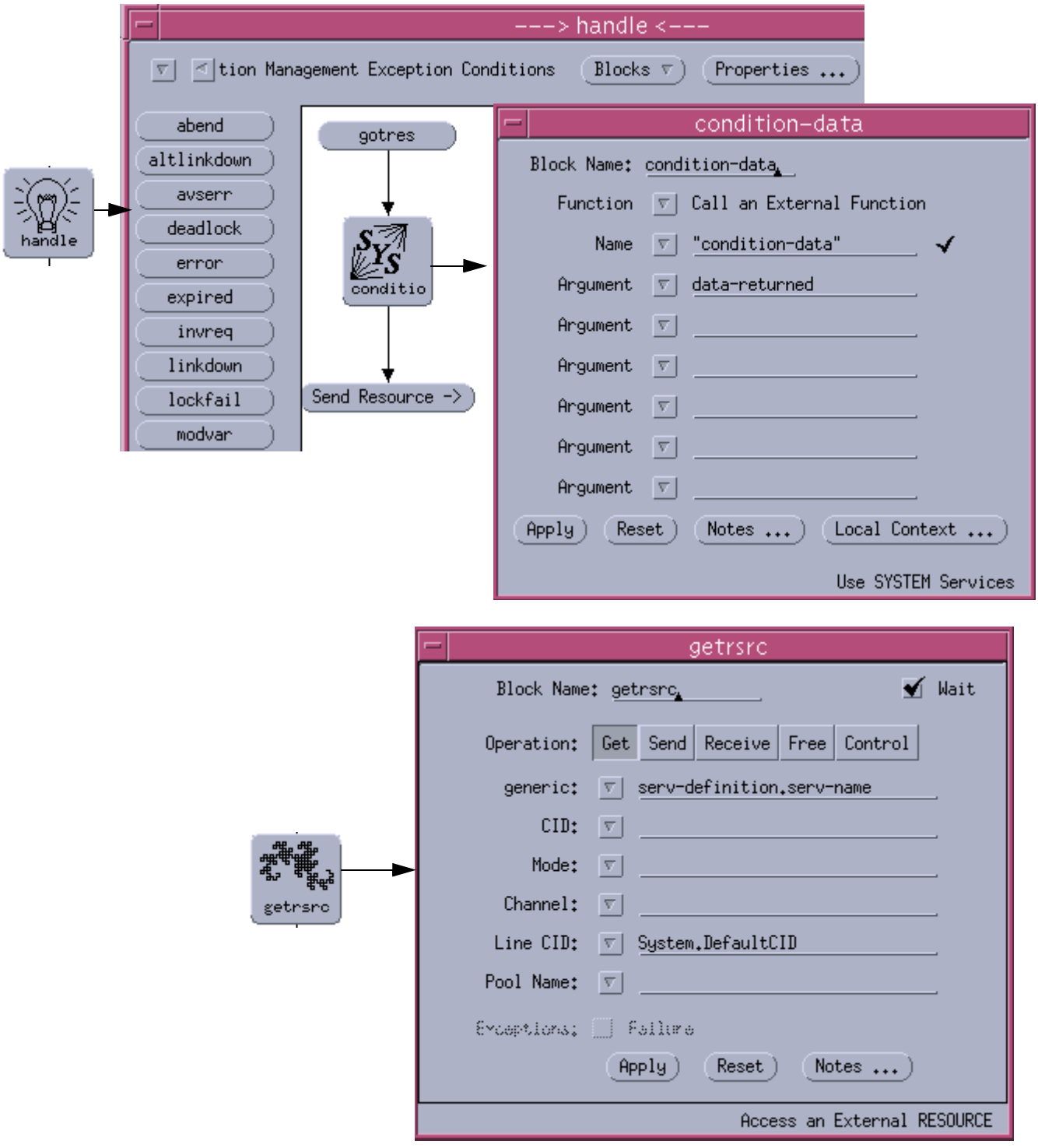
Ambiguity can be avoided if the VTCPD daemons use different names or if individual port numbers are specified in the PRS field. If several VTCPD daemons are running simultaneously, then each VTCPD must have a unique name. The unique VTCPD names must be specified in the `$ASEHOME/etc/services` file and by the command line option **-n** name.

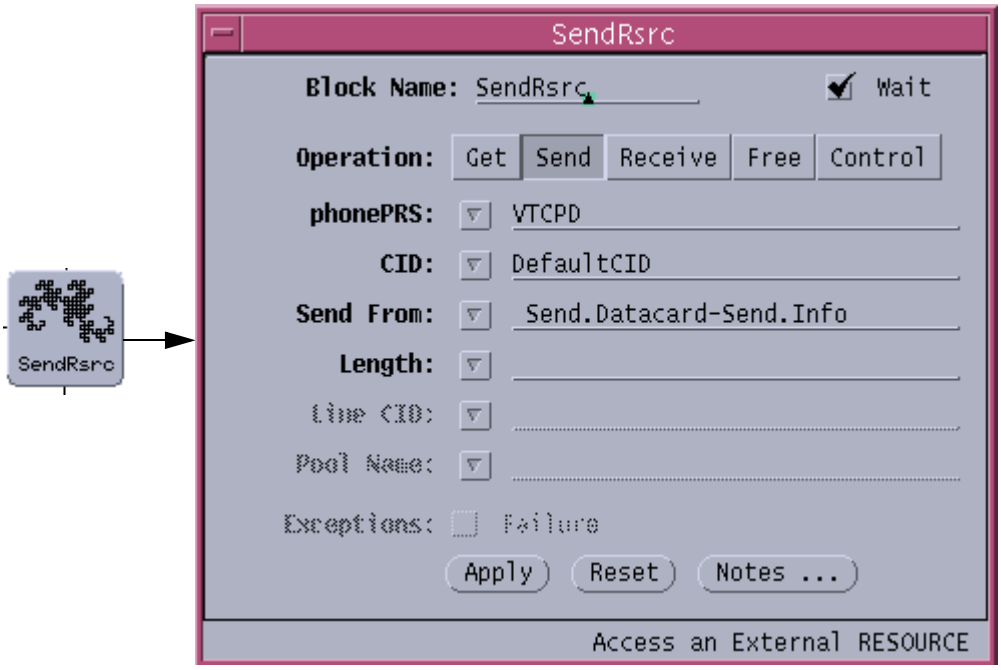
The application obtains the daemon port number from the data portion of **gotres** condition, which is returned for **GET** request. The format of the data portion is XXX:YYY:ZZZ, where X, Y and Z are connection number, slot number and daemon's port respectively. The port can then be used in the PRS field.
Example for the VTCPD daemon:

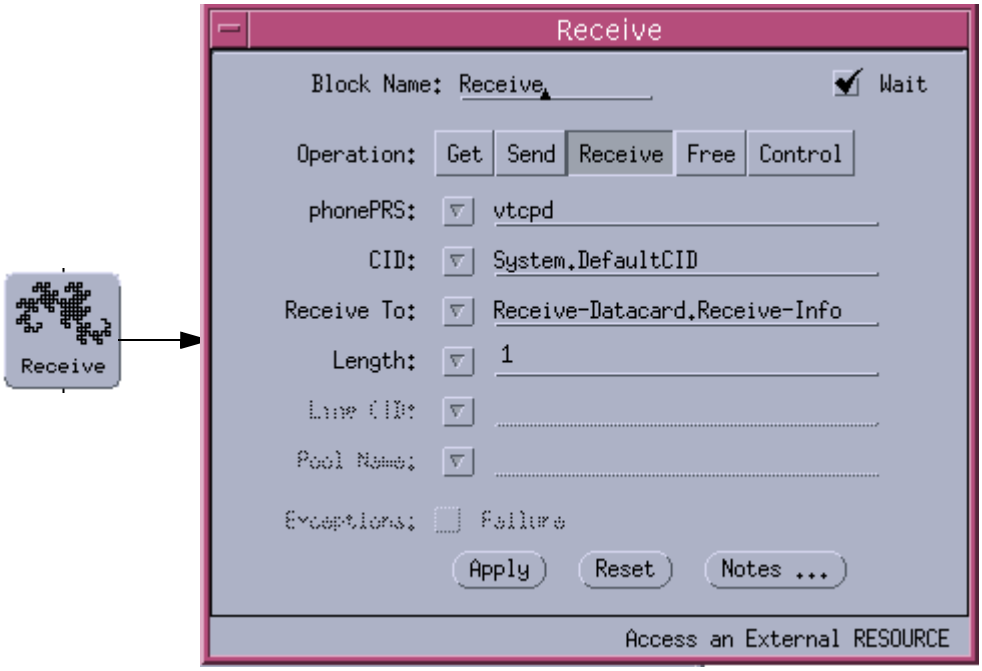
```
vtcpd -s 155 -n hlink1...
```

The sample PeriProducer application:




---> handle <---
tion Management Exception Conditions
Blocks
Properties ...
abend
altlinkdown
avserr
deadlock
error
expired
invreq
linkdown
lockfail
modvar
handle
gotres
conditio
Send Resource ->
condition-data
Block Name: condition-data
Function
Call an External Function
Name
"condition-data"
Argument
data-returned
Argument
Argument
Argument
Argument
Argument
Apply
Reset
Notes ...
Local Context ...
Use SYSTEM Services
getrsrc
Block Name: getrsrc
Wait
Operation:
Get
Send
Receive
Free
Control
generic:
serv-definition.serv-name
CID:
Mode:
Channel:
Line CID:
System.DefaultCID
Pool Name:
Exceptions:
Failure
Apply
Reset
Notes ...
Access an External RESOURCE
getrsrc





Releasing Connection Slot and Handling Exceptions

In addition to the situations mentioned above, the allocated slot is released and is made available for another phone line/request if:

- A new **SEND** request is received by the daemon and the slot has been acquired by the previous **SEND**.

- A new **GET** operation is received by the daemon.
- The application unbinds from VMST (exits or crashes).
- Host connection is lost. In this case, all applications currently using the link will receive condition hostdown.

An application can also receive a condition:

- **hostdown**, if at the time of the request there are no connections to the host at all, or if the specified connection is not established (or lost).
- **hostup**, if command option **-m u** is specified and the first link to the host is established. It allows applications to block on **DELAY** waiting for a **hostup** condition instead of looping indefinitely when host computer goes down. This condition is not returned to applications (by default).
- **getfail**, if specified pool or common pool is full. Optionally, all requests may be queued by the VTCPD daemon until there is an available slot.
- **invreq**, if **RECEIVE** request is sent prior to **SEND** (that is, no message ID is available) and the pool size is greater then 1 (there is no Message ID matching if the pools size is 1).
- **invreq**, if the specified connection doesn't fall into (1-maxlinks) range
- **invreq**, if the string in **SEND** message is not long enough to accommodate the Message ID.
- **inf**, if the string received from the host is not large enough to accommodate the message ID.
- **hostfail**, if the daemon cannot allocate memory or is experiencing some other internal problems.

4

VTCPD Application Programming

This chapter covers:

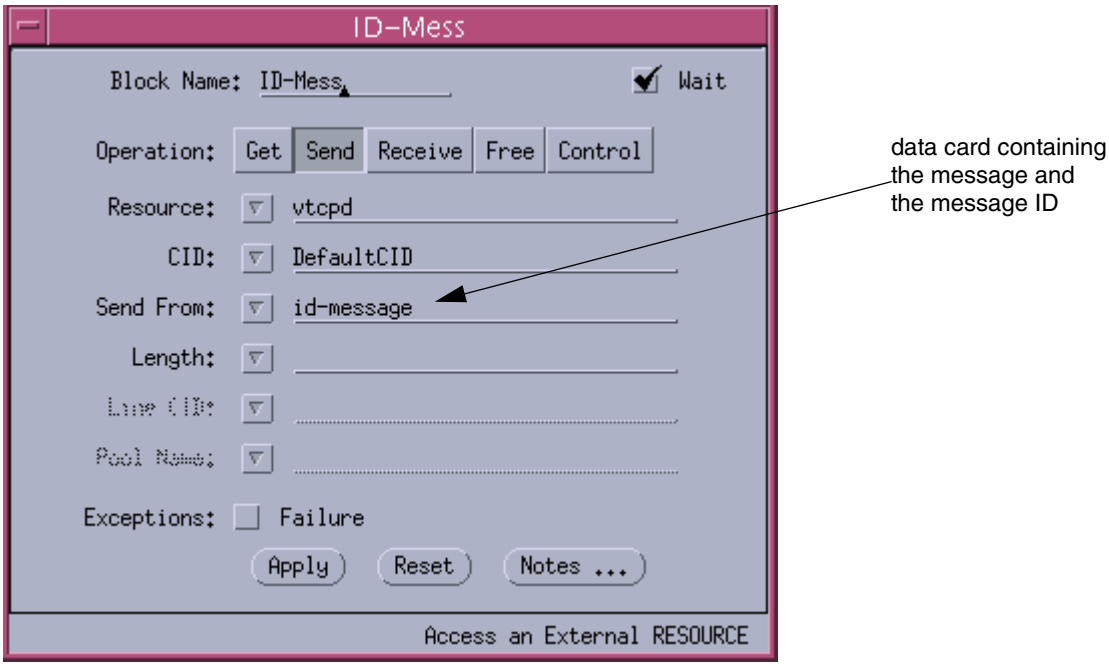
1. **Communicating with the Host**
2. **Establishing or Changing the Host Specifications**
3. **Specifying the Host Connection**
4. **Sending Messages**
5. **Receiving Messages**
6. **Configurations that Require Multiple VTCPD**
7. **Freeing the Host Connection**
8. **VTCPD Message Timing**

Communicating With the Host

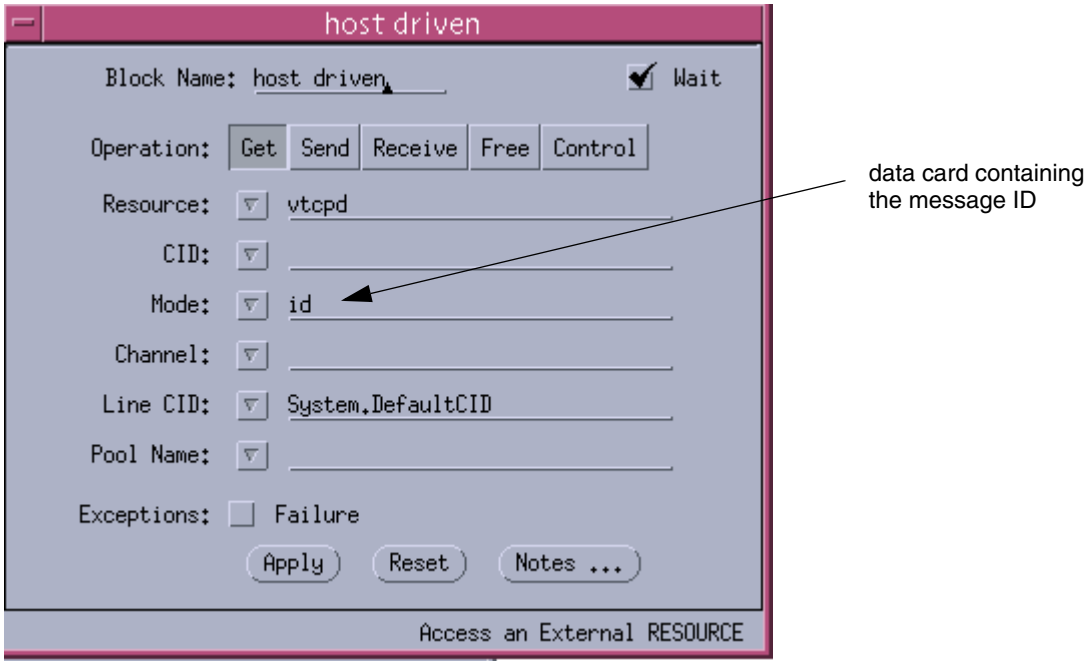
VTCPD uses the message ID to route host messages and application responses.

VTCPD can be configured to establish the message ID (**`vtcpd -I N:M or vtcpd -I N@string`**), or to allow the applications to establish the message ID (**`vtcpd -i N:M`** or **`vtcpd -i N@string`**).

To establish the message ID from an application, include the message ID, based on the length and starting position or string specified in the VTCPD configuration, in the Resource block that contacts the host. For example:



A host driven application (see ***Host Driven Applications*** on page 76.) can establish the message ID in the Resource block that contacts VTCPD. For example:



If the applications set the message ID, the applications must ensure that no two requests use the same message ID — in this configuration, VTCPD does not verify the message ID. If two outstanding requests use the same message ID, the results are unpredictable.

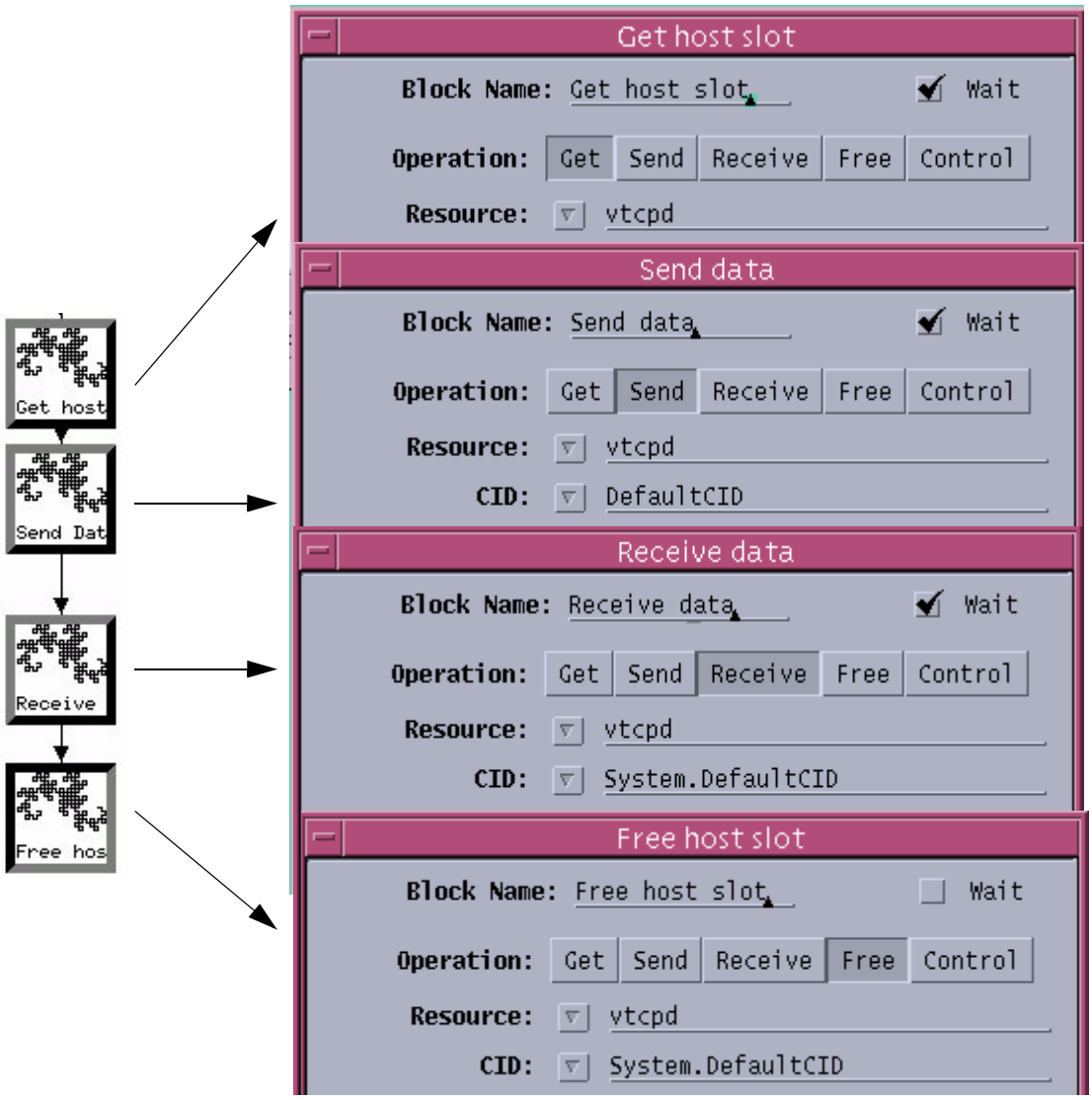
VTCPD can be started when some or all of the host specifications are unavailable. However, applications can not use these host connections until additional information is associated with the links. An administrative application can be used to establish the remaining host information (see ***Establishing or Changing the Host Specifications*** on page 79.)

The applications can be set up to reserve a connection and then solicit data from the host (see ***Soliciting Host Data*** on page 73.), or to wait for a host to send data (see ***Host Driven Applications*** on page 76.)

For configurations that support multiple message formats or systems that use a backup VTCPD, the application must be set to communicate with the appropriate instance of VTCPD (see ***Configurations That Require Multiple VTCPD*** on page 91.)

Soliciting Host Data

The application can be set to solicit host data (see ***Application-Defined Mode*** on page 29.)



In Get Host Slot, the application instructs VTCPD to reserve the host connection slot, see ***Backup LAN*** on page 38 and ***Specifying the Host Connection*** on page 81.

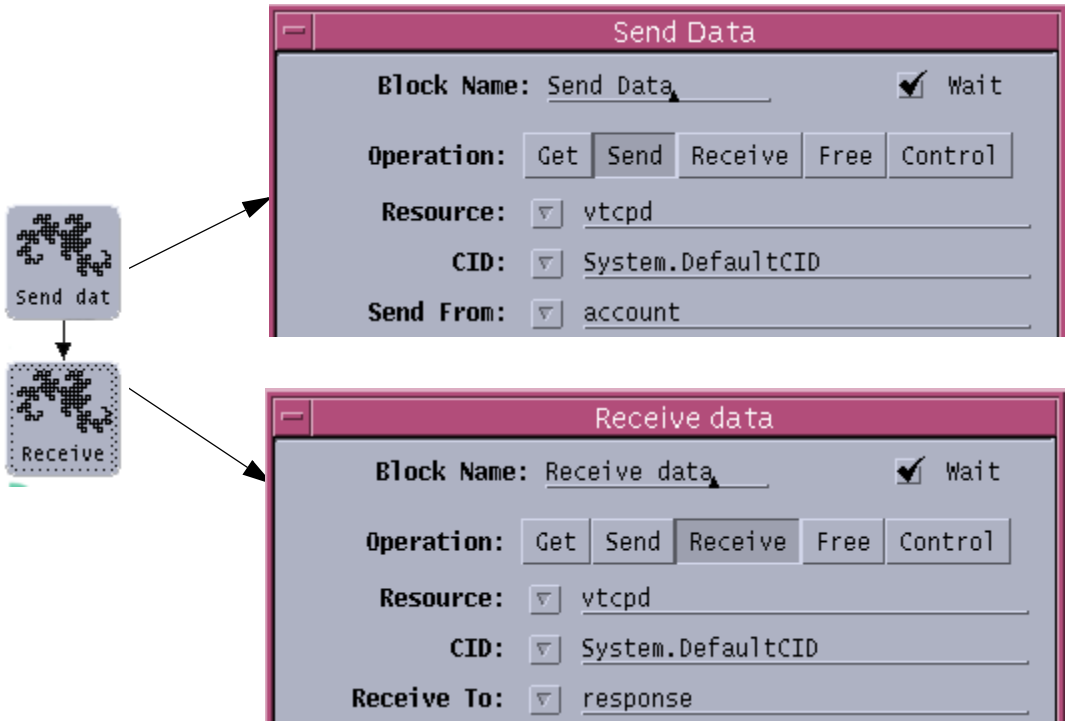
In Send Data, the application sends data to the host. This associates the message ID with the phone line. The message ID remains associated with the phone line until the application sends a message with a different ID, or the application releases the host connection slot (see ***Sending Messages*** on page 84.)

In Receive Data, the host uses the message ID to respond to the application. After associating the message ID with the phone line, the application can receive several messages (see ***Receiving Messages*** on page 87.)

In Free Host Slot, the application releases the host connection slot (see ***Freeing the Host Connection*** on page 95.)
In this standard scenario, the application reserves the host connection slot for the entire transaction. The application can be set to reserve the slot for a shorter time (see

Soliciting Host Data on page 73.)

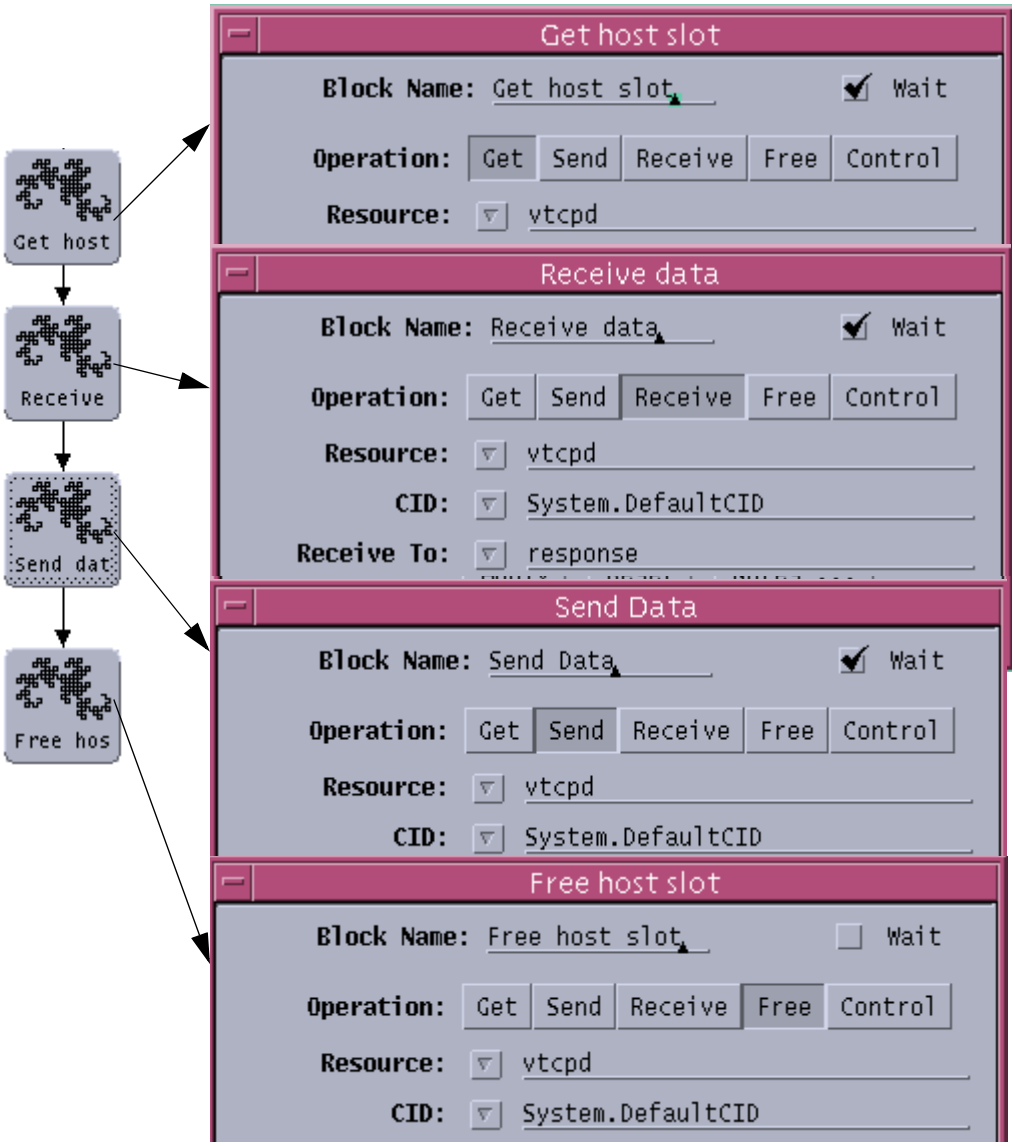
An application that solicits data from the host can reserve and release a host connection. However, if there is a limited pool of slots, the application can limit the time it reserves the slot by sending and receiving data without reserving and releasing the host connection (see ***Releasing Connection Slot and Handling Exceptions*** on page 69.)



VTCPD releases the slot after the application receives the host data.

Host Driven Applications

A typical host driven application waits for unsolicited data from the host (see *Application-Host Options* on page 36.)



In Get Host Slot, the application instructs VTCPD that it is available (see ***Specifying the Host Connection*** on page 81.) The application can wait for a particular message ID (see ***Receiving Data From a Known ID*** on page 77) or receive the next message that is not associated with another phone line (see ***Receiving the Next Available Host Message*** on page 78.)

In Receive Data, the application waits for data from the host. After associating message ID with the phone line, the application can receive several messages (see ***Receiving Messages*** on page 87.)

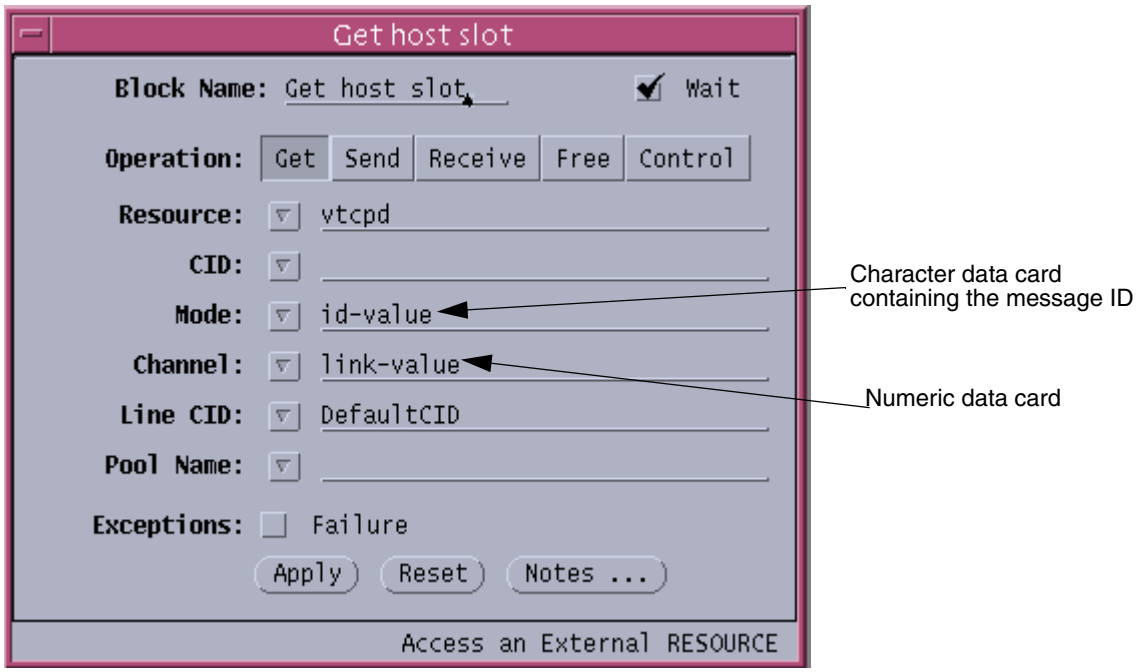
In Send Data, the application sends data to the host. The application can send additional messages, as long as the messages do not change the message ID (see ***Sending Messages*** on page 84.)

In Free Host Slot, the application releases the host connection slot (see ***Freeing the Host Connection*** on page 95.)

Receiving Data From a Known ID

To cause a host driven application to wait for data with a particular message ID, establish the message ID in the Resource block that contacts VTCPD (see ***ISSUE-RECEIVE-SEND (IRS)*** on page 64.)

For example:



After specifying the message ID, the application should use a Resource block to receive a message from the host.

Receiving the Next Available Host Message

The host driven application can be set to wait for the next message that is not associated with a particular line (see ***Inverted Mode*** on page 64.)

To support this interaction, VTCPD must be running in inverted mode—that is, with the command line options **`-m i`**.

The application must use Get Resource block to inform VTCPD that it is available:

When VTCPD receives the host data, it first tries to locate a phone line with the matching message ID. If there is no phone line with a matching message ID, VTCPD associates the message ID with the phone line running the available application. The application should use a Resource block to receive the message from the host (see ***Receiving Messages*** on page 87.)

After VTCPD associates the message ID with the phone line, the application can send and receive messages using the same message ID.

This Resource block signals all VTCPD attached to the VMST. If there is more than one VTCPD attached to the VMST, the particular VTCPD must be specified (see ***Configurations That Require Multiple VTCPD*** on page 91.)

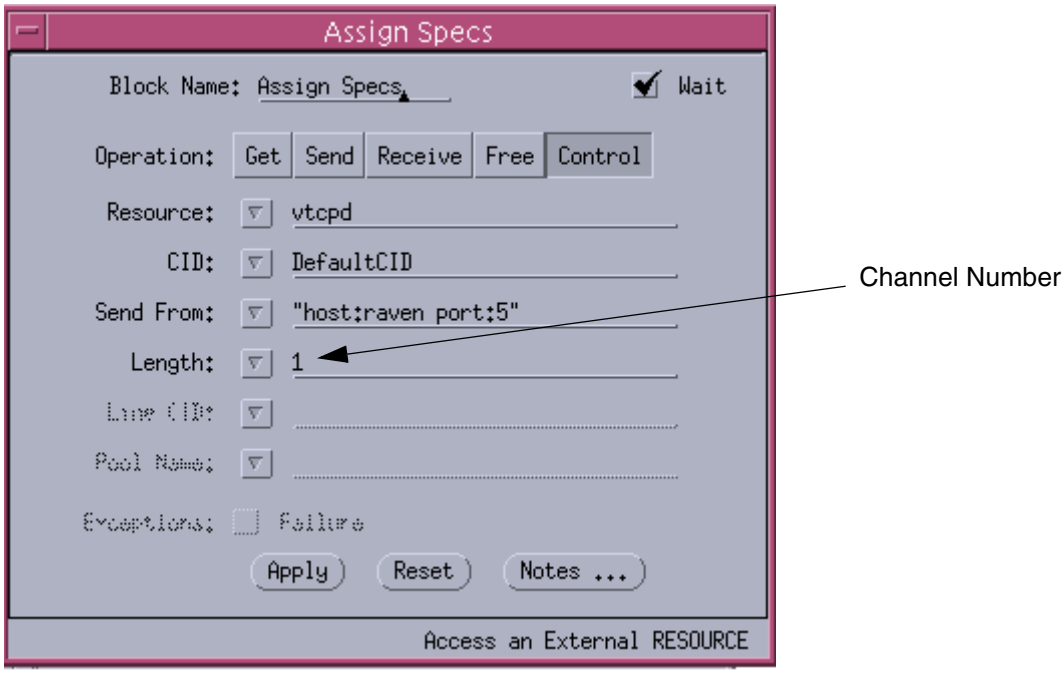
Establishing or Changing the Host Specifications

The system can start a VTCPD when all or some of the host specifications are unavailable (see ***Application-Defined Mode*** on page 29.) However, applications cannot use these host connections until additional information is associated with the links. The Resource block can be used to establish the remaining host information also to dynamically change the host specifications.

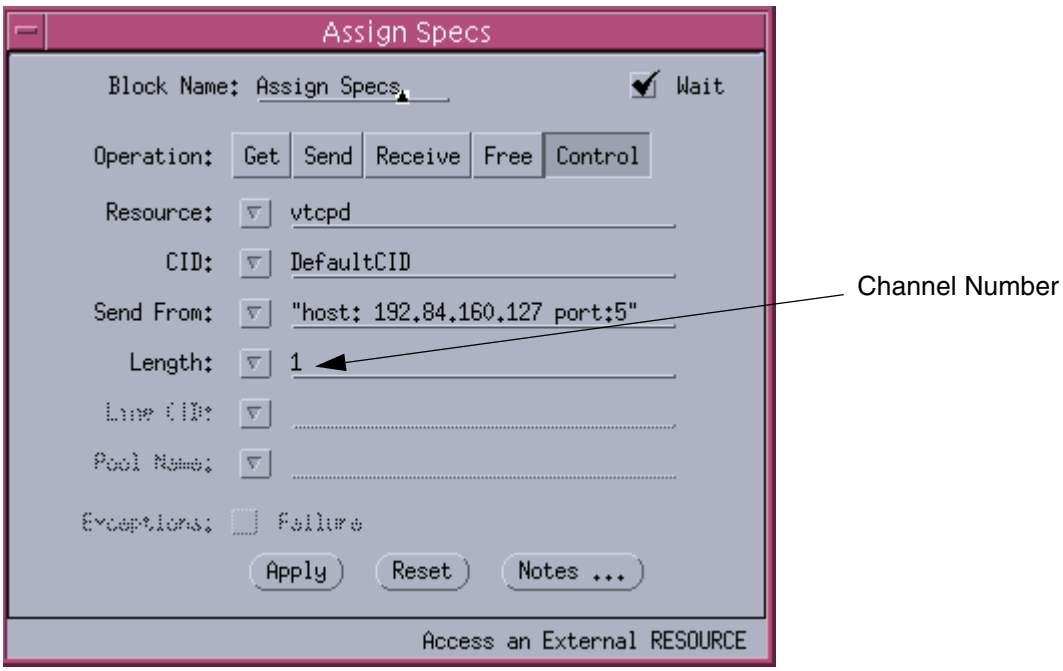
Use a data card or a literal in the Send From field (Mode field in PeriProducer version 2.3) to set the host name, the port number, or both. Use the syntax `host:<hname> port:<number>`. The host can be specified using either the host's name or the its IP address.

The Length field (Channel field in PeriProducer 2.3) can be used to specify the connection number. The connection number must be in the range from one to the total number of host connections. The total number of host connections is defined by the VTCPD configuration (**`vtcpd -l #`**) (see ***Host Connection Options*** on page 26.) If the specified link is connected, VTCPD closes the link before making the connection. This feature can be used to instruct VTCPD to reestablish a connection — if, for example, an administrative application misses pings from the host.

For example, to set the host to `raven` and the port to 5:



To set the host to the IP address 192.84.160.127:



Specifying the Host Connection

VTCPD can be configured to pool requests for each connection to a host, provide a common pool for all requests, or allow unlimited requests from any connection (see ***Releasing Connection Slot and Handling Exceptions*** on page 69.)

The application can be set to specify a host connection (see ***Specifying the Connection From the Application*** on page 81), or allow VTCPD to specify the host connection (see ***Specifying the Connection From the Application*** on page 81.)

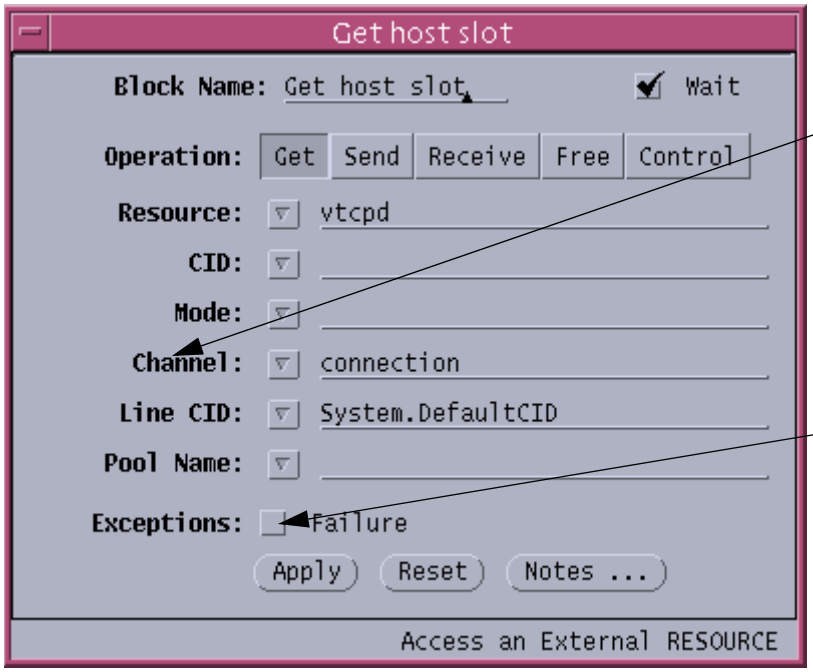
To send and receive messages on a system configured for multiple VTCPD, the application must specify the particular instance of VTCPD.The condition data associated with **`gotres`** contains the connection number, the slot number, and VTCPD port number. The application can use this information to communicate with the correct VTCPD. For additional information, see ***Configurations That Require Multiple VTCPD*** on page 91.

Specifying the Connection From the Application

Use a Resource block to request a host connection. For VTCPD configured for a particular host, the Channel field can be used to specify the connection number (see ***Application-Host Options*** on page 36.)

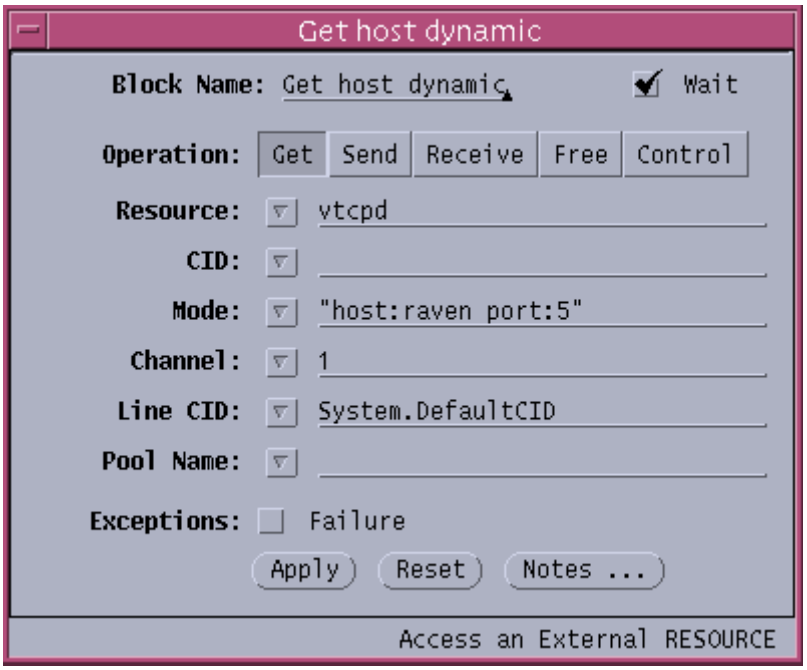
The connection number must be in the range from one to the total number of host connections. The total number of host connections is defined by the VTCPD configuration (**`vtcpd -l #`**) (see ***Host Connection Options*** on page 26.)

For example:



For VTCPD configured without host specifications (that is, *without* **`vtcpd -l`**

options), the application can request the connection dynamically.
For example, to request a connection to port 5 on the host named `raven`:

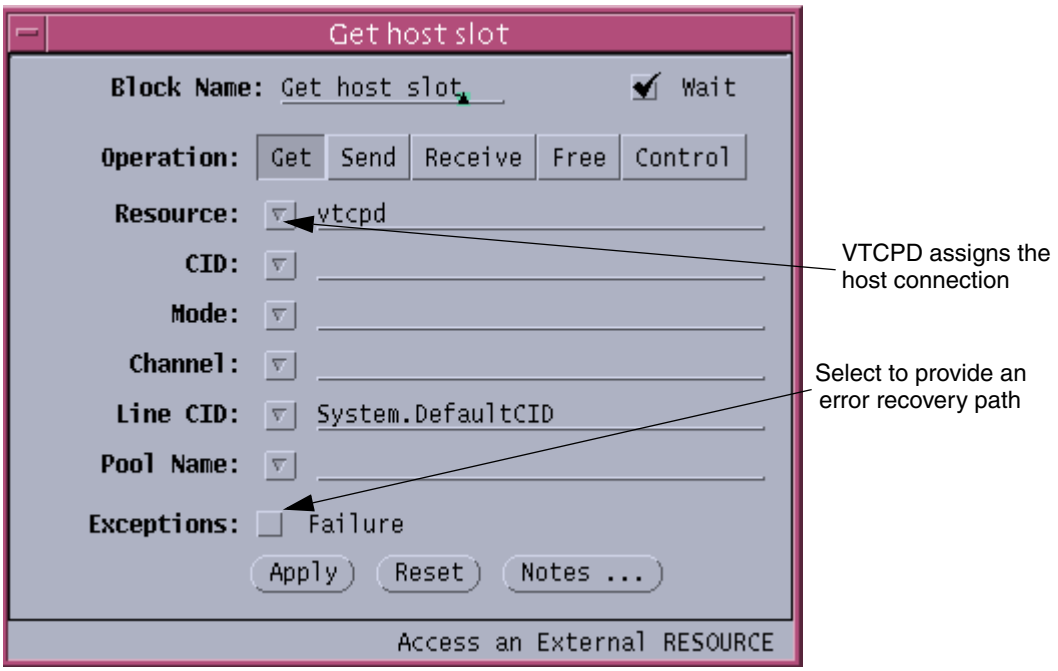


If the specified connection is full, the application receives **`getfail`**. Select Failure in the Resource block to provide an error recovery path (see ***VTCPD Status and Exception Conditions Reference*** on page 106.) A Handle block can also be used to provide error recovery. If the application gets the requested connection, it receives **`gotres`**.

If the application terminates, VTCPD frees the host connection (see ***Freeing the Host Connection*** on page 95.)

Allowing VTCPD to Specify the Connection

Use a Resource block to instruct VTCPD to assign a host connection (see ***Application Connection Options*** on page 35.)



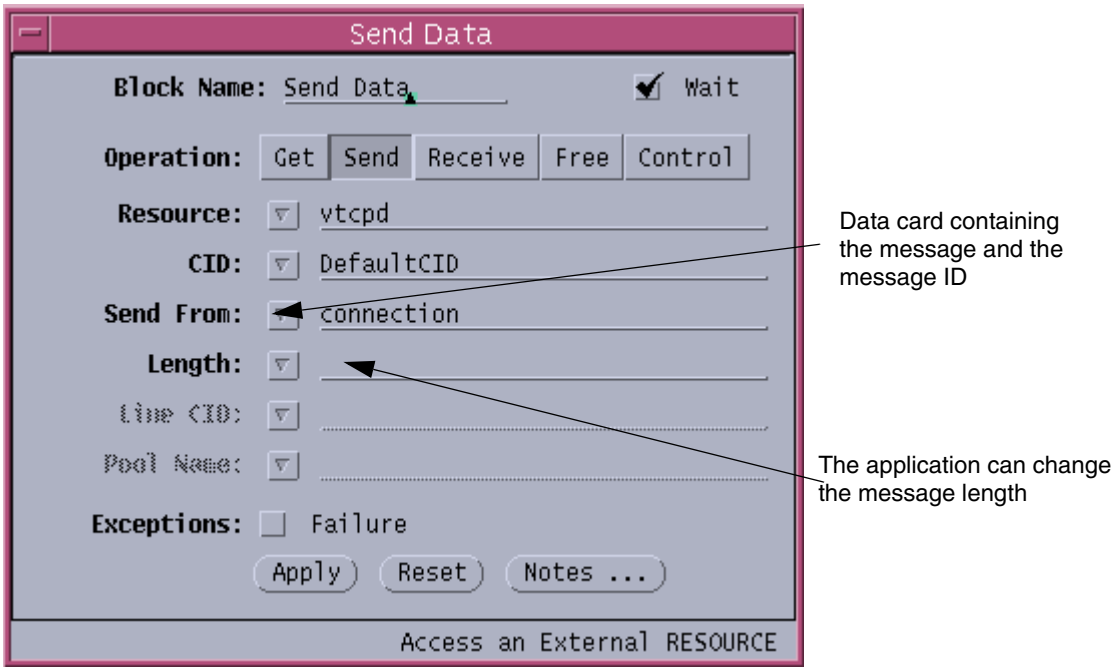
Since the application does not specify a channel, VTCPD chooses the connection and the slot in the pool using round robin load balancing (see ***Message Routing*** on page 53.)

If all the connections are full, the application receives **`getfail`** (see ***VTCPD Status and Exception Conditions Reference*** on page 106.) Select Failure in the Resource block to provide an error recovery path. A Handle block can also be used to provide error recovery. If the application gets a host connection, it receives **`gotres`**.

If the application terminates, VTCPD frees the host connection (see ***Freeing the Host Connection*** on page 95.)

Sending Messages

Use Resource blocks to send messages to the host.

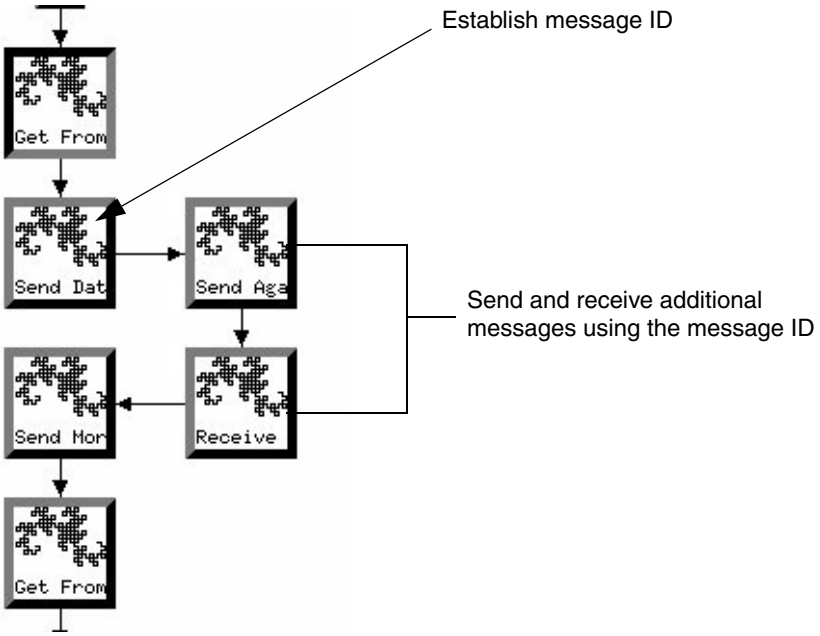


In most cases, VTCPD uses the message ID to route messages. Any request from the application to the host and any response from the host to the application must contain the same message ID.

The VTCPD can be configured to establish the message ID, or allow the application to establish the message ID.

In an application that solicits data from the host, the first Send establishes the message ID (see ***Soliciting Host Data*** on page 73.) A host driven application can establish the message ID in the Resource block that contacts VTCPD (that is, the Resource block with the Get operation) or wait for VTCPD to establish the message ID (see ***Host Driven Applications*** on page 76.)

After the message ID is established, the application and the host can continue to send and receive messages using multiple Resource blocks. For example:



To send and receive messages on a system configured for multiple VTCPD, the application must specify the particular instance of VTCPD. For additional information, see ***Configurations That Require Multiple VTCPD*** on page 91.

Routing Replies in a Host Driven Application

If an application only receives one host message at a time, VTCPD can be configured to route the application replies through the same connection as the host request (**`vtcpd -ml`**) (see ***Message Routing*** on page 53.)

However, if the application can receive additional messages from other hosts before it is done with processing the first message, and the application must reply through the same connection as the host request, VTCPD must be configured to include the link number in the messages from each host (**`vtcpd -mL`**). In this configuration, VTCPD adds twelve ASCII bytes to the front of each message from a host. These bytes are in the format

`XXX:YYY:ZZZ:`

where `XXX` is the link number (`YYY` and `ZZZ` are reserved for future use). For VTCPD to correctly route the reply, the application must add these twelve bytes to the front of each reply.

Variable Length Messages

VTCPD can be configured to support either fixed length messages or variable length messages (see ***Message Format*** on page 46.)

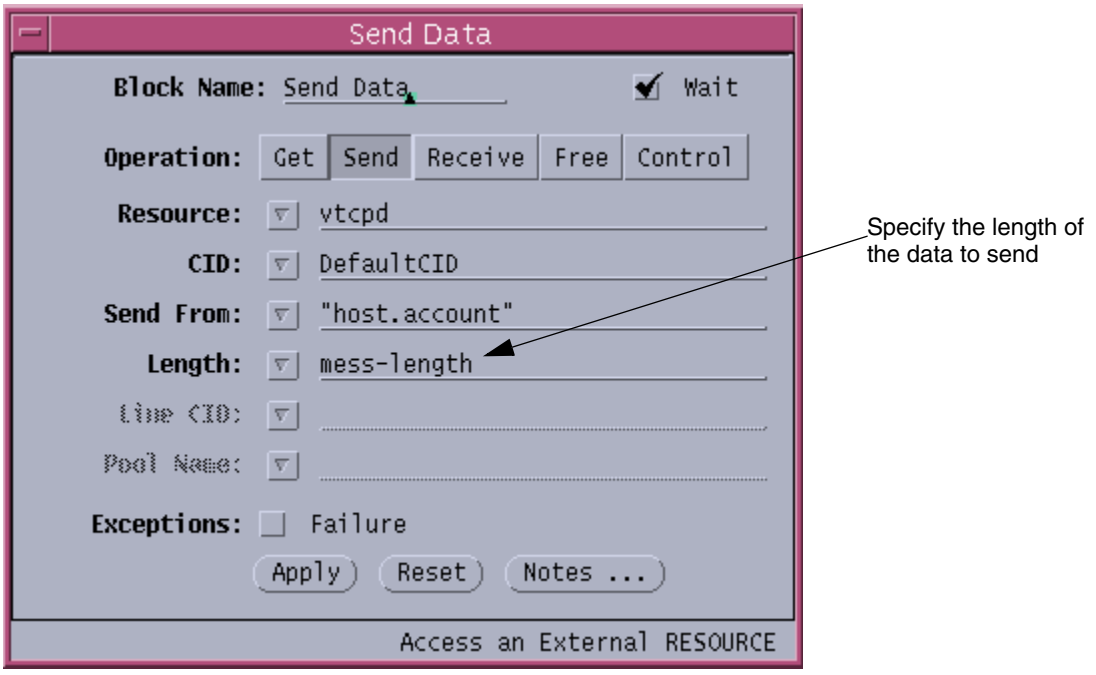
Based on the VTCPD configuration, a variable length message is either delimited by a special character or specified by a length header that precedes the message (the length header can be included in the total length of the message).

VTCPD is responsible for supplying the delimiter or the length header. *The application should only send data the delimiter or length header is added by VTCPD.*

The application can also specify the length of the message that it sends to the host (see ***Variable Length Messages*** on page 86.)

Applications can use the Resource block Length field to specify the length of the message sent to the host (*see* ***Message Format*** on page 46.) *The length of the message must not be less than the length of the message ID plus the position of the message ID.* This feature can be used if, for example, the data changes dynamically but the application stores it in the same data card.

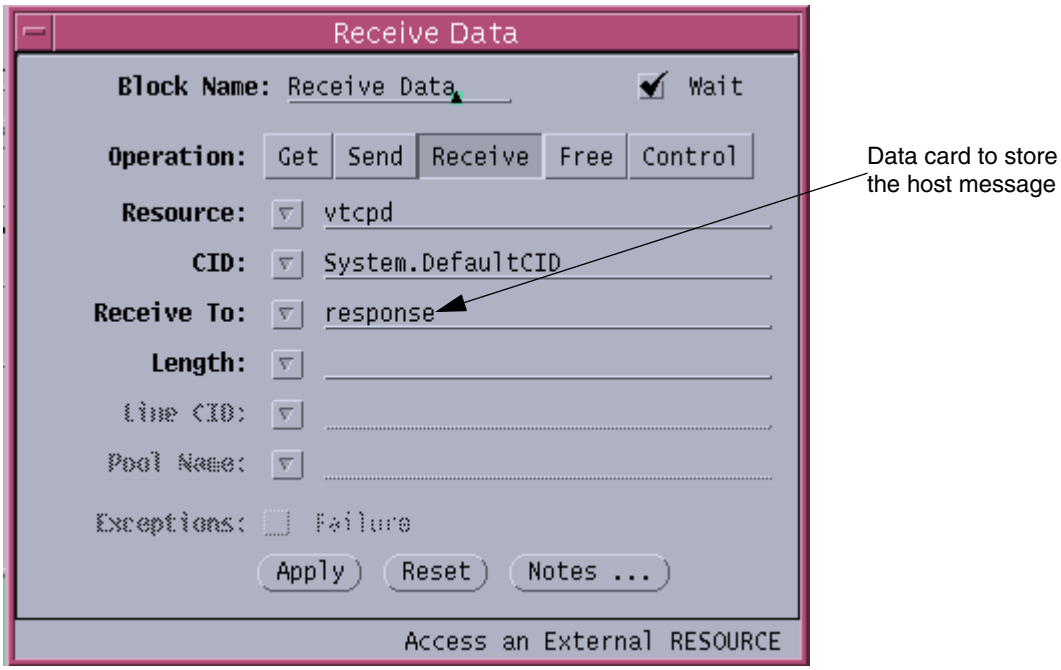
The application cannot change the length of a message if VTCPD is configured for fixed length messages (that is, **`vtcpd -f <len>`**).



If the application specifies the length of the message, VTCPD ignores the truncate option, **`vtcpd -mt`** (see ***Message Format*** on page 46.)

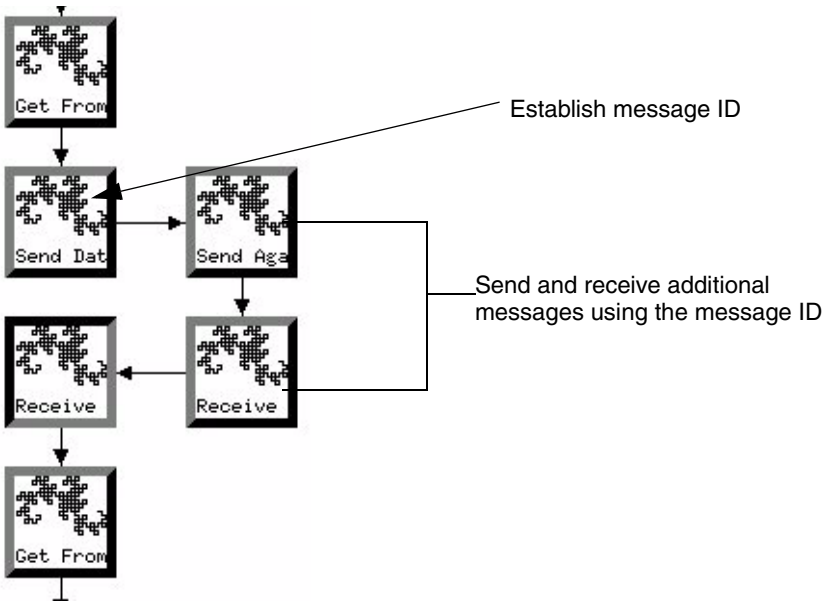
Receiving Messages

Use Resource blocks to receive messages from the host (see ***Replies Routing*** on page 60.)



In most cases, VTCPD uses the message ID to route messages. Any request from the application to the host and any response from the host to the application must contain the same message ID. The host response normally arrives through the connection the application used to send the request. VTCPD can also be configured to route host responses from any connection (see ***Host Responses From Any Connection*** on page 89.)

After the message ID is established, the application and the host can continue to send and receive messages using multiple Resource blocks. For example:



If the host sends a message before the application executes a Resource block to receive the message, VTCPD holds the host message. VTCPD can also be configured to send messages asynchronously (see ***Asynchronous Replies and Reply Notification*** on page 89.)

To avoid possible race conditions, applications should ignore both **`intertimeout`** and **`ertimeout`** after receiving the host message (see ***VTCPD Message Timing*** on page 96.)

Receiving Message Headers

VTCPD supports either fixed length messages or variable length messages delimited by a character or preceded by a header (see ***Message Format*** on page 46.)

In addition, VTCPD can be configured to remove (that is, strip) the header before it sends the message to the application, or to send the entire message (including the header).

If VTCPD is configured to send the header with the message, the application can be set to parse (or process in some other manner) this header information. For information about using composite folders to parse data, see the *PeriProducer User s Guide*.

For messages sent *from* the application, VTCPD *always* supplies the delimiter or header.

Host Responses From Any Connection

The host response normally arrives through the connection the application used to send the request. VTCPD can also be configured to route host responses from any connection (**`vtcpd -md`**) (see ***Message Routing*** on page 53.)

If VTCPD is configured to route host responses from any connection, internal message processing is delayed slightly.

Asynchronous Replies and Reply Notification

Normally, if the host message is not available when an application executes a Resource block to receive a message, the application pauses (blocks). However, VTCPD can be configured for asynchronous data delivery (**`vtcpd -m a`**) (see ***Asynchronous Replies and Reply Notification*** on page 89.)

If VTCPD is configured for asynchronous data delivery and the application has not already executed the Resource block to receive the message, VTCPD sends the message to the application as condition data associated with **`unexdata`**. The application can retrieve the data from the System.ConditionData data card.

VTCPD can also be configured to notify the application when the host sends a message (**`vtcpd -m n`**). When the host message arrives, VTCPD notifies the application by generating **`unexdata`**. In this configuration, when the application receives **`unexdata`** it should execute a Resource block to receive the message.

Retrieving Unidentified Host Data

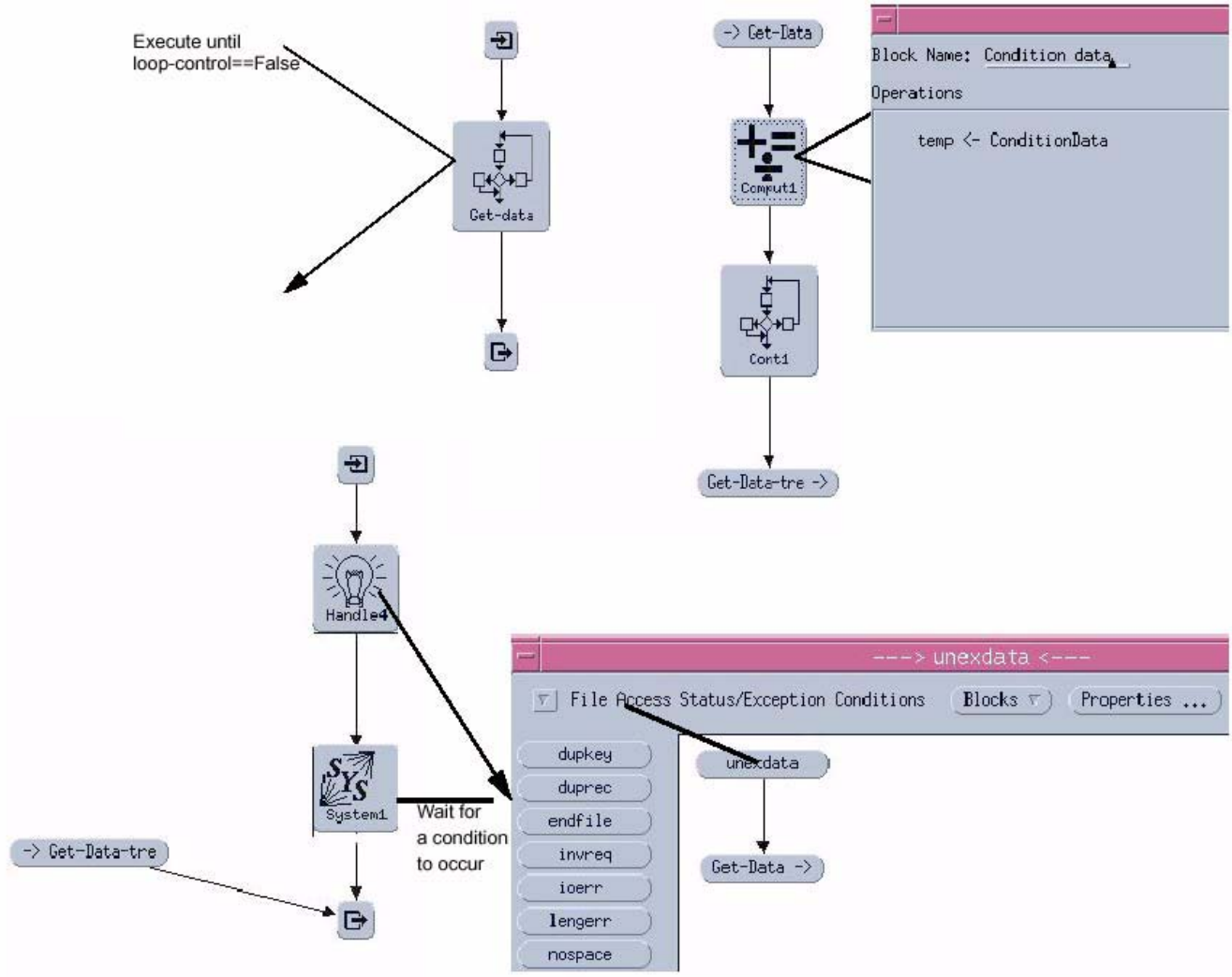
Normally, if the host data contains a message ID that VTCPD cannot associate with a phone line, VTCPD discards the message. However, VTCPD can be configured to route all unidentified host data to a particular phone line that is running an administrative application (**`vtcpd -a#`**) (see ***Unidentified Host Data*** on page 62.)

In this configuration, VTCPD sends the unidentified messages to the administrative line as condition data associated with **`unexdata`**. The application can retrieve the data from the `System.ConditionData` data card.

There are several important considerations for an administrative application designed to retrieve unidentified host messages:

- VTCPD sends these message asynchronously. To avoid possible race conditions, *the administrative application should not execute any blocks that pause the execution of the application*.
- In most circumstances, the administrative application should not restart. If the application restarts, it can lose messages that arrive during the transition period. However, to properly reconnect to VMST, the application must restart if it receives **`linkdown`**.

The following demonstrates the implementation of an administrative application:


Execute until
loop-control==False
Get-data
-> Get-Data
Block Name: Condition data
Operations
temp <- ConditionData
Comput1
Cont1
Get-Data-tre ->
Handle4
System1
Wait for
a condition
to occur
-> Get-Data-tre
---> unexdata <---
File Access Status/Exception Conditions
Blocks
Properties ...
dupkey
duprec
endfile
invreq
ioerr
lengerr
nospace
unexdata
Get-Data ->

Configurations That Require Multiple VTCPD

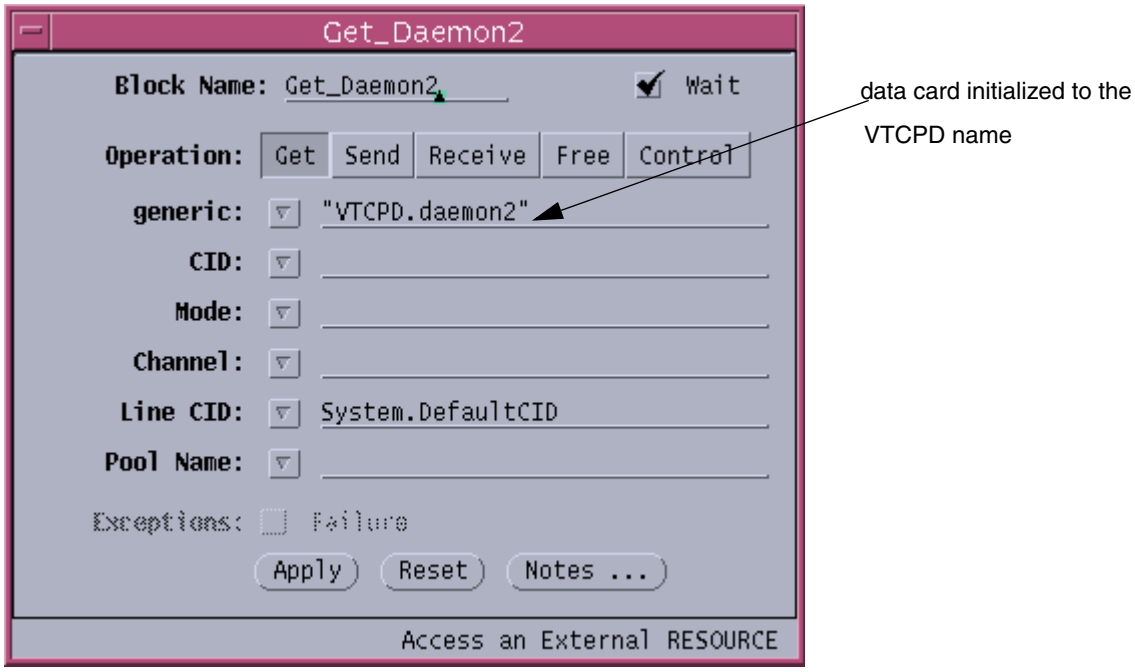
Each instance of VTCPD supports a single message format. If the Avaya Media Processing Server must communicate with TCP/IP hosts using multiple formats, the system must be configured with an instance of VTCPD for each format.

By default, Resource blocks that contact VTCPD broadcast requests to all VTCPD attached to the same VMST. If the system uses multiple VTCPD, the application must specify the particular instance of VTCPD (see ***Communicating With the Host*** on page 72.)

Specifying Which VTCPD

If the system supports multiple VTCPD for multiple message formats, each VTCPD is configured to with a unique name. The VTCPD names are specified in `$ASEHOME/etc/services`. The command **`vtcpd -n <name>`** is used to specify the VTCPD (see ***Attaching to VMST*** on page 34.)

Specify the VTCPD name in the Resource block that contacts VTCPD. For example:



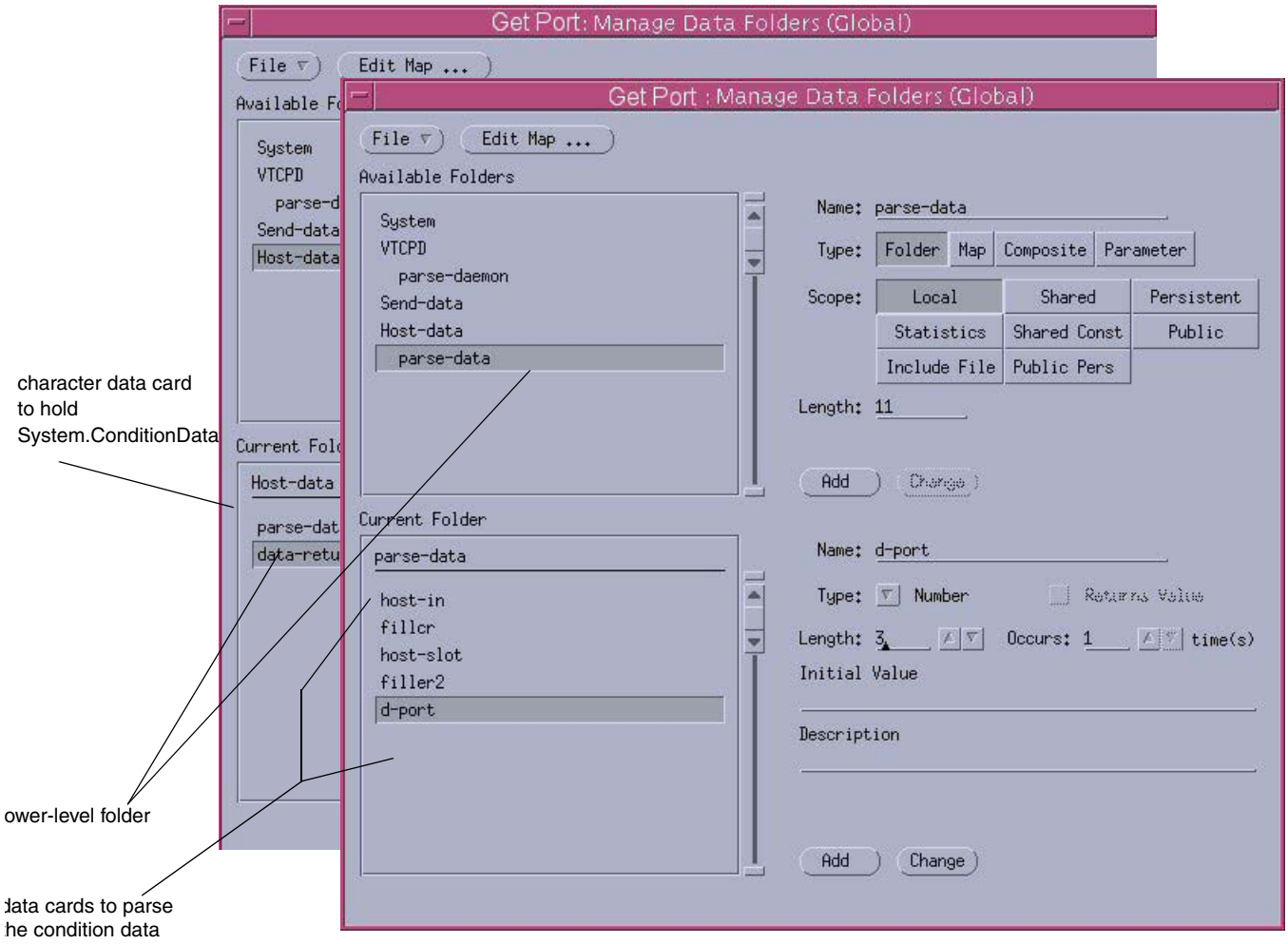
Using the VTCPD Port Number

VTCPD returns the port number as condition data associated with **`gotres`**. The format of the condition data is XXX:YYY:ZZZ where XXX is the connection number, YYY the slot number, and ZZZ the port number.

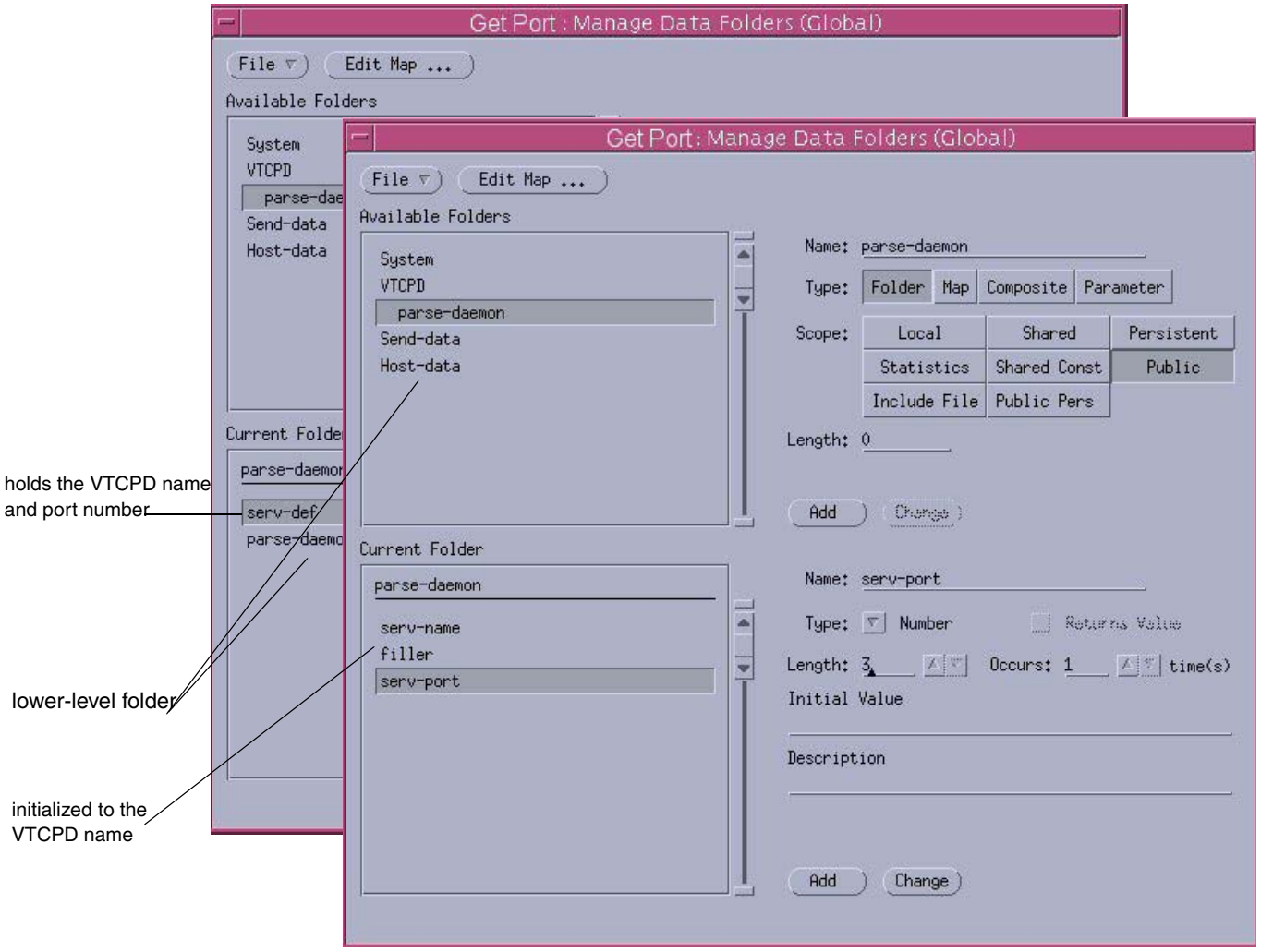
The following demonstrates an application that uses the port number to send data to the appropriate VTCPD.

Data Setup

To parse the condition data, this application uses a composite folder that contains both a character data card to store the condition data and a lower-level folder. Data cards in the lower-level folder parse the data.

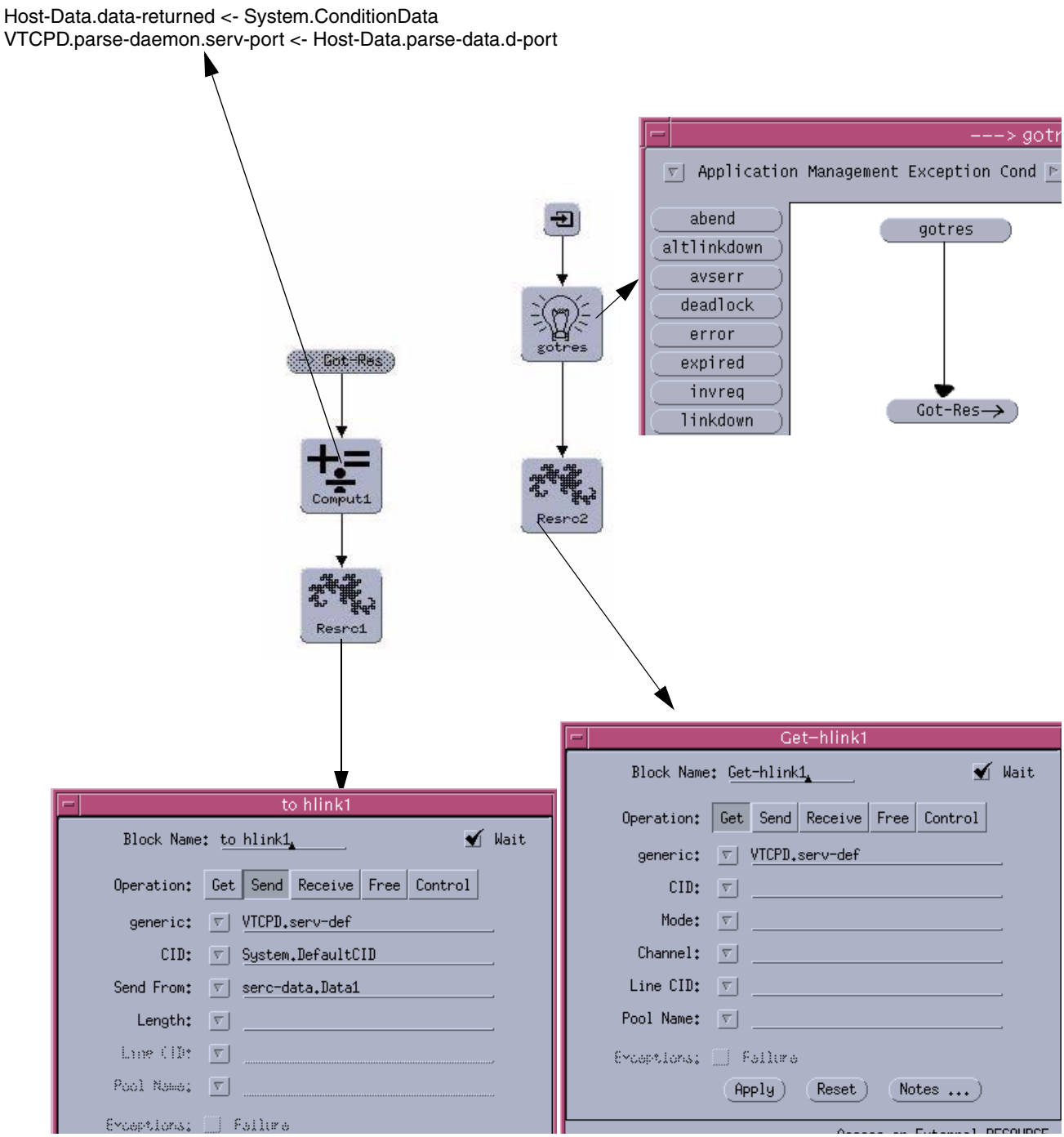


To communicate with VTCPD, this application uses a character data card initialized to the VTCPD name. Using a lower-level folder, this data card also reserves space for the VTCPD port number.



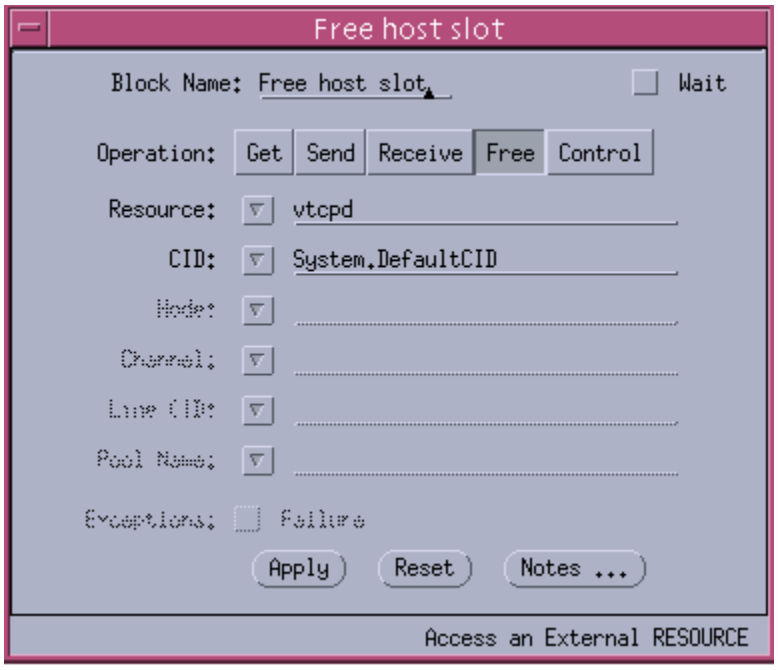
Implementation

First, this application uses `VTCPD.serv-def`, initialized to the VTCPD name, to contact VTCPD. When the application receives **`gotres`**, it parses the condition data to determine the port number. Then the application uses port number to send a message to the specific instance of VTCPD.



Freeing the Host Connection

Use a Resource block to free the host connection slot (see ***Releasing Connection Slot and Handling Exceptions*** on page 69.)



To send and receive messages on a system is configured for multiple VTCPD, the application must specify the particular instance of VTCPD. The condition data associated with **`gotres`** contains the connection number, the slot number, and VTCPD port number. The application can use this information to communicate with the correct VTCPD. For additional information, see ***Configurations That Require Multiple VTCPD*** on page 91.

VTCPD also releases the host slot in the following circumstances:

- The application disconnects (that is, executes a Disconnect block or a Main Container Exit connector)
- The application sends a message containing a different message ID
- The application requests a different connection (see ***Specifying the Host Connection*** on page 81.)
- The system loses the host connection (in this case, applications using the connection receive **`hostdown`**)

If the application sends and receives data without explicitly reserving and releasing the host connection, VTCPD releases the host slot after the application receives the host data. For additional information, see ***Soliciting Host Data*** on page 73.

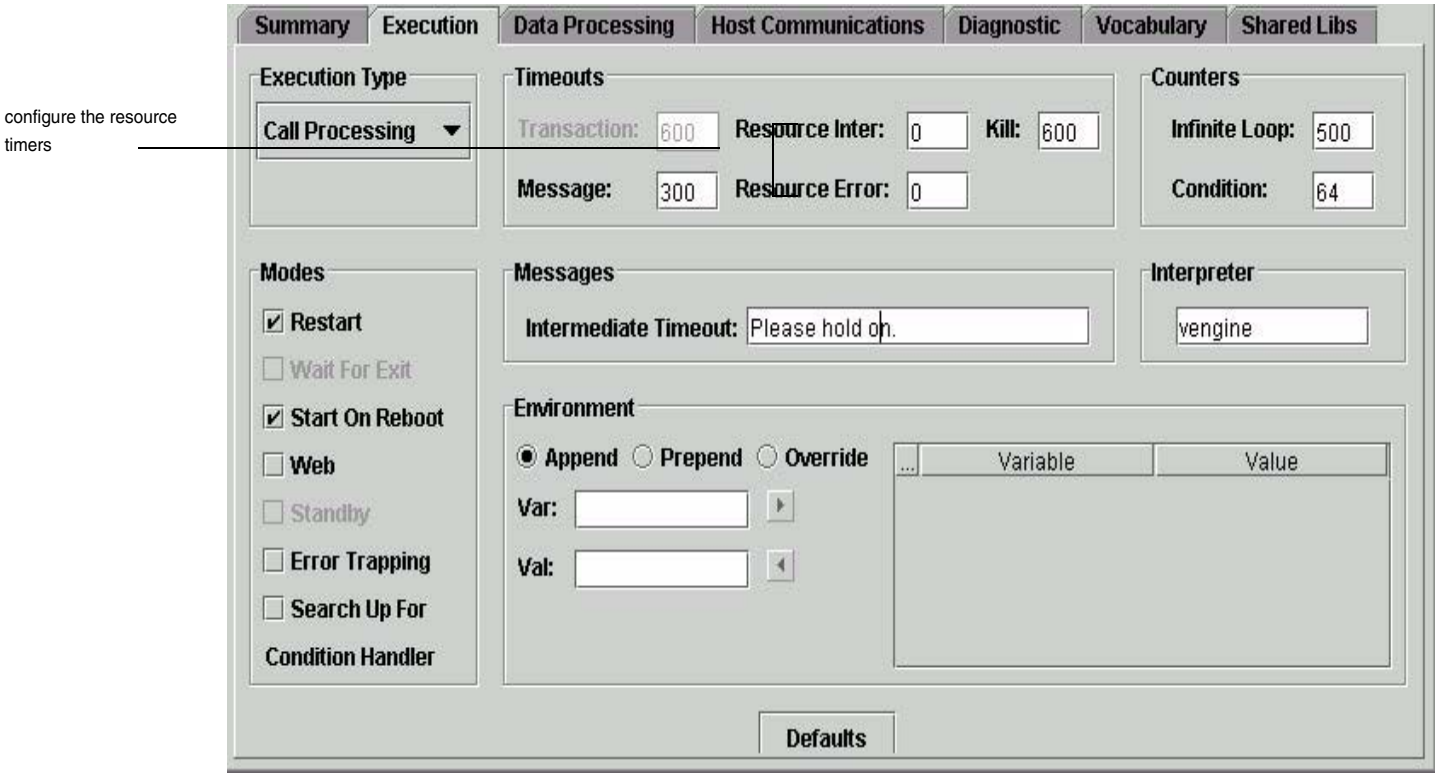
VTCPD Message Timing

The VENGINE can be configured to maintain a resource intermediate timer and a total resource error timer for VTCPD message requests (see ***Time-Outs*** on page 39.)

Both the resource intermediate timer and the resource error timer are disabled by default.

VENGINE starts timing when an application requests data from the host. When the intermediate timer expires, the application speaks a “Please hold on” message and VENGINE restarts the intermediate timer. When the error timer expires, VENGINE sends **`ertimeout`** to the application (see ***VTCPD Status and Exception Conditions Reference*** on page 106.)

To configure the resource intermediate timer and the resource error timer, use the Application Manager Configure Applications tool. In the Execution Options window, use the Resource Inter field to set the intermediate timer and the Resource Error field to set the error timer. For details about the Configure Applications tool, see the *PeriView Reference Manual*.

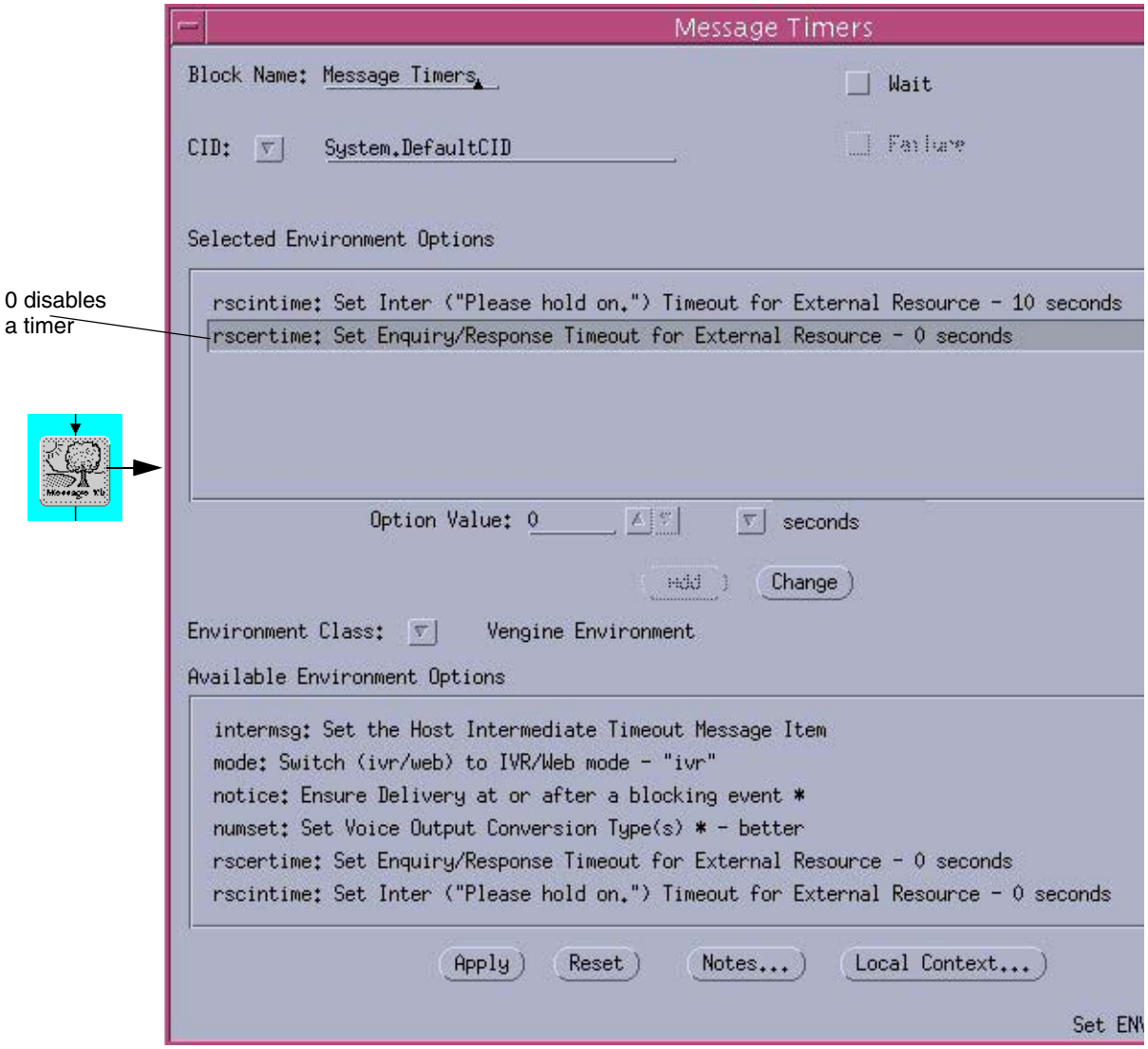


These timers can be enabled using the VENGINE command line option **`-Q<ito>:<eto>`** where **`<ito>`** and **`<eto>`** are the values in seconds (0 disables a timer).

The applications can be used to set these message timers dynamically, see ***Setting Resource Timers from an Application*** on page 97, and ***Handling Resource Timeouts*** on page 97.

Setting Resource Timers from an Application

Use Environment blocks to modify resource message timing within an application (for details, see the *PeriView Reference Manual*).

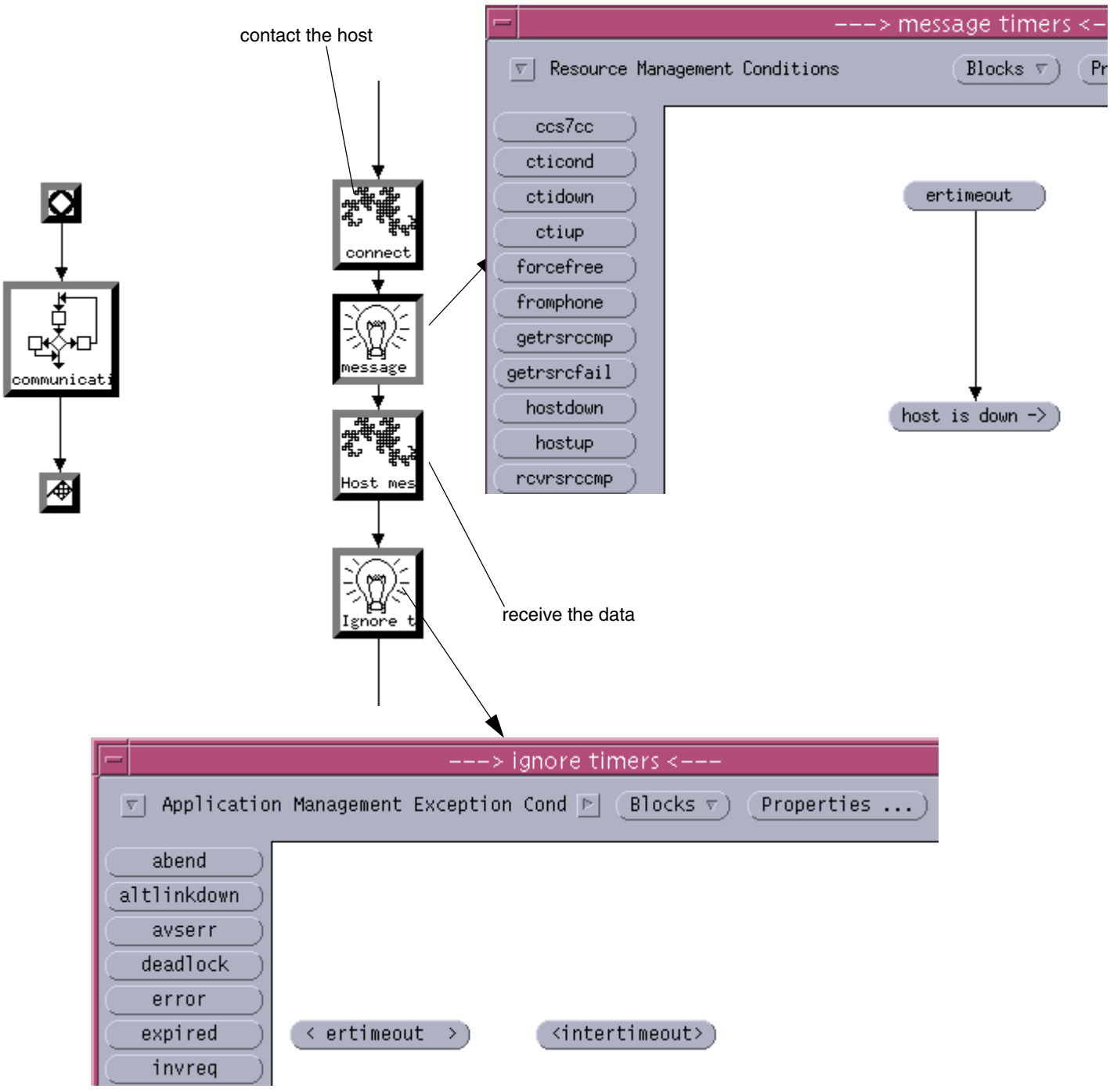

0 disables a timer
Message Timers
Block Name: Message Timers
Wait
CID: System.DefaultCID
Failure
Selected Environment Options
rscintime: Set Inter ("Please hold on.") Timeout for External Resource - 10 seconds
rscertime: Set Enquiry/Response Timeout for External Resource - 0 seconds
Option Value: 0
seconds
Add
Change
Environment Class:
Vengine Environment
Available Environment Options
intermsg: Set the Host Intermediate Timeout Message Item
mode: Switch (ivr/web) to IVR/Web mode - "ivr"
notice: Ensure Delivery at or after a blocking event *
numset: Set Voice Output Conversion Type(s) * - better
rscertime: Set Enquiry/Response Timeout for External Resource - 0 seconds
rscintime: Set Inter ("Please hold on.") Timeout for External Resource - 0 seconds
Apply
Reset
Notes...
Local Context...
Set ENV

Handling Resource Timeouts

By default, when an application receives **`intertimeout`**, the application speaks a "Please hold on" message to the caller. To customize this message, use the Application Manager Configure Applications tool; modify the Host Timeout Message in the Host Communications Options window (for details, see the *PeriView Reference Manual*). The VENGINE can also be used to customize this message using command line option **`-o`**.

When an application receives **ertimeout**, the application should explain to the caller that the host is currently unavailable, or possibly refer the caller to a live operator.

When an application receives data from the host, the system cancels all pending message time-outs. However, it is possible to generate **intertimeout** or **ertimeout** in the instant before the application receives the data. In this case, the application can receive **intertimeout** or **ertimeout** after the application receives the host data. The *application must ignore **intertimeout** and **ertimeout** after receiving data from the host*.


contact the host
---> message timers <-
Resource Management Conditions
Blocks
ccs7cc
cticond
ctidown
ctiup
forcefree
fromphone
getrsrccmp
getrsrcfail
hostdown
hostup
rcvrsrccmp
ertimeout
host is down ->
communicati
connect
message
Host mes
Ignore t
receive the data
---> ignore timers <---
Application Management Exception Cond
Blocks
Properties ...
abend
altlinkdown
avserr
deadlock
error
expired
invreq
< ertimeout >
<intertimeout>

This page has been intentionally left blank.

5

VTCPD Debugging and Maintenance

This chapter covers:

1. **Debugging**
2. **Fault Tolerance**
3. **VTCPD Status and Exception Conditions Reference**
4. **VTCPD Troubleshooting**

Debugging

The current status of the internal VTCPD tables can be obtained by the **amu** command request status. It shows the state of host connections, pool slots allocated to each phone line, current line state (**I -** the line sent **GET; S - SEND; R -RECEIVE**, waiting for host reply; **P -** data from host pending, waiting the line to execute **RECEIVE**) and associated Message ID or data (if pending).

To facilitate debugging:

- display a VTCPD status report
  The traffic between the VTCPD daemon and applications can be displayed if the command option **-Xa** has been used or if it is enabled by the **amu** command request debug a. Use **req deb n** to disable the feature (see ***amu request status*** on page 102.)

- display message traffic
  The traffic between the daemon and host(s) can be displayed by command line option **-X1** (or **req deb 1** and **req deb 0** using **amu** commands) (see ***amu request debug*** on page 102.)

- simulate host input
  The host may be simulated using an instance of VTCPD and one or more PERIProducer applications (see ***Host Simulation*** on page 103.)

Displaying Debugging Information

To display the current status of the internal VTCPD tables, enter the command

amu request status

This report shows:

- the state of host connections
- the pool slots allocated to each phone line
- the current line state (**I** - requested a host connection, **S** - sent data to the host, **R** - waiting for a host message, **P** - (pending) holding the host data until the application requests the data)
- the message ID or the entire host message if VTCPD is holding the host message (pending)

amu request debug

Display message traffic between VTCPD, the application(s), and the host(s):

To display message traffic between VTCPD and the application(s), enter the command:

```
amu request debug a
```

To disable this feature, enter the command:

```
amu request debug n.
```

To display message traffic between VTCPD and the host(s), enter the command

```
amu request debug 1
```

To disable this feature, enter the command:

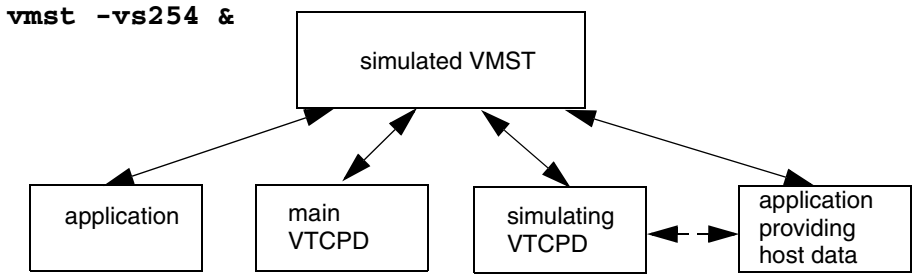
```
amu request debug 0.
```

VTCPD can be configured to display message traffic. For configuration information, see the *Avaya Media Processing Server Series System Reference Manual*.

Host Simulation

For testing and debugging or if the host is temporarily unavailable, the host may be simulated using an instance of VTCPD and one or more PERIProducer applications. The simulator has to be started in the inverted and server modes if the main VTCPD is running in direct and client modes and vice versa. The simulating VTCPD delivers all host messages to the PERIProducer application and the PERIProducer application sends the messages that would normally come from the host.

For this purposes both the VTCPD daemon and the application can be attached to the simulating VMST, started for example:



If the main VTCPD is running in direct and client mode, the simulating VTCPD must be running in inverted and server mode; if the main VTCPD is running in inverted and server mode, the simulating VTCPD must be running in direct and client mode. For information about configuring VTCPD, see the *Avaya Media Processing Server Series System Reference Manual*.

To simplify development of the host side processes sample code is included to deal with network messaging. The source can be found in the `$ASEHOME/etc/vtcpd.host.sample file`.

Fault Tolerance

Daemon Backup
A simple way to automatically restart the daemon in case of a crash is to invoke the VTCPD using the following script:

```
#!/bin/csh
while (1)
 vtcpd -v mpsname -l ...# Keep it in foreground
end # Will be reached only if
    # vtcpd crashes
```

Monitoring Host Connections
If a host process dies, the kernel on the host machine notifies the VTCPD, which will, in turn, send a condition **`hostdown`** to the applications. However the VTCPD will not be notified if the entire host machine crashes or if the LAN connection is broken. To detect this situation, the VTCPD should query the host periodically with a ping command, declaring that it is down if no replies arrive in specified time.

Pinging the Host from the VTCPD
To query the host with a ping command, the timeout period must be specified by the option -T timeout (in seconds). By default (and if the -h option is enabled) the ping message is sent every three seconds which can be changed by the second part of **`-T`** option: **`-T 30:10`** generates ping message each 10 seconds with total timeout of 30 seconds.

It is also necessary to declare the format of the ping message by the **`-h length[:string`**] option. For message formats using delimiters (**`-f d:ch`**) or length headers (**`-f 1:#`**) the length of the ping message may equal to zero (in which case only the header or delimiter is shipped to the host. The string part of the option **`-h`** is ignored). For fixed length format the length of the ping message can not be zero though it may be smaller then the length of the message specified by -f len option.

If the string part is not specified, the ping message is filled with ASCII '0'. If the string contains exactly one character, the message will be filled with this character: -h4 and -h4:0 and -h4:0000 are equivalent.

To specify an unprintable character, the hexadecimal representation (0xXX) should be used. For example: **`-h 4:0x00`** defines the ping message of 4 binary zeroes.

If the length of the string (each hexadecimal representation is counted for one character) is greater then one, it should be exactly equal to the specified length: **`-h 4:A0x410x42B`** represents AABB message.

Pinging the VTCPD from the Host

In cases where ping messages are initiated by hosts, the messages have to be rerouted to administrative lines (VTCPD option **-a** or **-A**) for processing and reply. If an administrative application determines that a host link went down (for example, if there is no traffic for a certain period of time), it can force the VTCPD to break the host connection. This can be achieved by sending to the VTCPD an AMU-style command:

```
amu -s XXX -v YYY -c "request break ZZZ".
```

Here XXX, YYY and ZZZ are VTCPD port numbers (as specified by VTCPD -s option), MPS number and connection number respectively. This command could be sent from the application by **execute-command** call function.

Backup LAN

It is possible to have backup LAN connection between the MPS and host machines. The VTCPD daemon will try to switch to this LAN if all host connections through the primary LAN are lost. To specify the feature the use the command line option:

```
-L [#:]host:port
```

The host is the host machine name as it is known on the secondary network. Parameters # and port have the same meaning as for **-l** option. Several **-L** options can be used to specify different hosts/port numbers.

VTCPD Status and Exception Conditions Reference

The following table lists the status and exception conditions and their messages.

Status and Exception Conditions Messages

Error Message	Description
ertimeout	The host did not respond to a VTCPD message request within the interval set by the enquiry/response timer. See ***VTCPD Message Timing*** on page 96.
getfail	The application could not get a VTCPD connection to the host — the specified pool or the common pool is full. See ***Specifying the Host Connection*** on page 81.
gotres	The application successfully connected to the host. Condition data associated with **gotres** contains the connection number, the slot number, and the port number. This data can be used to route messages (see ***Using the VTCPD Port Number*** on page 91).
hostdown	The application requested a host connection but VTCPD does not have any host connections. The application also receives **hostdown** if the host process terminates. If VTCPD is in UPD mode, the application does not receive **hostdown**. The application only receives **hostdown** if VTCPD receives a corrupted message from the UPD host.
hostfail	VTCPD can not allocate memory, or some other internal problem.
hostup	VTCPD can be configured to send **hostup** to the application when the host process restarts (**vtcpd -mu**). By default, applications ignore **hostup**. However, the applications can be set to wait for **hostup** rather than looping when the host goes down.
inf	The string from the host is not large enough for the message ID.
intertimeout	The host did not respond to a VTCPD message request within the interval set by the intermediate timer. The application speaks a “Please hold on” message and the intermediate timer restarts. See ***VTCPD Message Timing*** on page 96.
invreq	1) application requested a host message and VTCPD has not established a message ID. 2) application specified a connection that is not in the valid range 3) the application sent a message that is not long enough for the message ID — the length of the message must not be less than the length of the message ID plus the position of the message ID (see ***Sending Messages*** on page 84)
unexdata	VTCPD is configured for asynchronous data delivery or notification. See ***Asynchronous Replies and Reply Notification*** on page 89.

VTCPD Troubleshooting

The VTCPD Troubleshooting is composed of two parts. A listing of Typical Problems is provided to assist in resolving common problems, see ***Typical Problems*** on page 107. A Questions and Answers listing is also provided, again this is to assist in resolving common problems, see ***Questions and Answers*** on page 109.

Typical Problems

Check the listing of Typical Problems to see if your problem is listed.

- ***VTCPD cannot connect to the host:*** on page 107
- ***Data from host does not arrive to VTCPD:*** on page 107
- ***The reply from the host is not routed to the application:*** on page 108
- ***The host does not send correct data:*** on page 108
- ***Questions and Answers*** on page 109
- ***If nothing else helps:*** on page 108

VTCPD cannot connect to the host:

- Check that the host is defined in /etc./hosts and /etc./ethers VTCPD command line correctly specifies host name or IP address
- Make sure that:
  the host is up (use the **/usr/sbin/ping** command)

  the appropriate process on the host is running and is using correct TCP/IP port
- Use snoop, paying special attention to source and destination ports

Data from host does not arrive to VTCPD:

- Using VTCPD command line option **`-X2`**, check that data actually don't arrive from the host (arriving data are marked in debugging by << From Host)
- Verify in snoop that data are sent by the host and the destination port matches ephemeral port, assigned to VTCPD.
- Check that VTCPD is not in an infinite loop (send **`amu`**-command request status). If VTCPD has been started from your terminal you should see some output, otherwise truss the daemon. If you see infinite loop contact R&D.

The reply from the host is not routed to the application:

- Check that the ID field is specified correctly on VTCPD command line
- Enable VTCPD debugging using -**Xa** and **-X2** options
- Run snoop to trace IP packets
- Verify that the ID fields in the request and in the reply do match and are located in the proper place (-i or -I option)
- Verify that the message from the host is delimited according to **-f** option (see snoop output) and does not have extra bytes at the end, (which are not the beginning of the next message).

  Note the field left 000 in the VTCPD output:

  From Host Len total VTCPD MPS Msg Len left
  vtcpd [MPS 254, 09:02:26]: <<

  The field says that there were no extra bytes at the end of the message.
  BbcIP] PRE>

- Verify that the application did send a request for data (or that alternative options for async delivery are specified in the VTCPD command line)

The host does not send correct data:

- Either you have to change VTCPD's command line options to accommodate the host data format
- Or negotiate a data format change with the host developers

If nothing else helps:

- Install the newest version of VTCPD.
- Ask for R & D Assistance.

R & D provides assistance in complicated cases when the solution is not clear, or if you think that the new functionality or bug fix is needed. However, we need VTCPD trace and snoop output separated in 2 different files:

- **vtcpd -Xa -X2** (or debugging enabled by amu). The file must include VTCPD command line and output from following commands:

```
vrc
what `which vtcpd` |
grep vtcpd

snoop -x0 -ta ...
```

Questions and Answers

Check the listing of Questions and Answers to see if your problem is listed.

- ***Application intended to process all host replies (vtcpd `-A` option) is losing some messages.*** on page 109
- ***Two different types of messages (fixed length and variable length messages delimited by newline NL) need to be processed. How is this done?*** on page 109
- ***An application is host driven (the ID field is set by the host), how should the application be programed?*** on page 109
- ***There are two servers on the host side, but one only receives requests from applications and another one only replies.*** on page 110
- ***Conditions `intertimeout` or `ertimeout` sometimes arrive after the application receives data from the host, disrupting the execution.*** on page 110
- What does this message mean?***`VTCPD05 [MPS 7, PID 4067, port 133, Dec 13 16:41:08.230]: Got and closed UNNEEDED Host '148.185.40.22:3003' connection #2 [socket 13]`*** on page 110
- ***If nothing else helps:*** on page 108

- **Application intended to process all host replies (vtcpd `-A` option) is losing some messages.**

  An application can lose messages while restarting (all messages delivered while the program is in transition period will be discarded). To avoid this, program the application to return to the RECEIVE or DELAY operation as soon as it finishes to process the current message. However, this can trigger an infinite loop error, which should be disabled by VENGINE -l option.

- **Two different types of messages (fixed length and variable length messages delimited by newline NL) need to be processed. How is this done?**

  Start two VTCPD daemons with options **`-f len`** and **`-f d:0x0a`** respectively and name them (use two different names) by specifying the name option (**`-n`**). These new names have to be placed into `$ASEHOME/etc/services` file and the VMST must be restarted.

- **An application is host driven (the ID field is set by the host), how should the application be programed?**

  Use the inverted mode, specified by the option **`-m i`**. It allows the application to register by the **Get** operation in the PeriProducer resource block for receiving host data.

- **There are two servers on the host side, but one only receives requests from applications and another one only replies.**

  Open two connections and always use the first one for writing by specifying **`CHANNEL(1)`** in the **Get** resource block. The daemon command line option -m d will allow the application to receive replies from a different link:

  ```
  vtcpd -l server:9000 -l server:9001 -m d...
  ```

- **Conditions `intertimeout` or `ertimeout` sometimes arrive after the application receives data from the host, disrupting the execution.**

  Timing service is provided by the **`vastimer`**, in releases prior to 5.1.2. Because of this, the possibility of race conditions exists. To avoid it always, use

  ```
  IGNORE condition intertimeout ertimeout
  ```

  As soon as data is received from the host.

- **What does this message mean?**

  ```
  VTCPD05 [MPS 7, PID 4067, port 133, Dec 13
  16:41:08.230]: Got and closed UNNEEDED Host
  '148.185.40.22:3003' connection #2 [socket 13]
  ```

  VTCPD was started in a server mode and specified a certain number of connections to accept by **`-l`** option: **`vtcpd -nVTCPD05 -v7 -s133 -l 3003 -f 402 -p 1 -a 248.`** However, a connect to VTCPD was attempted and the number of open connections exceeded the specified maximum. Either increase the number by **`-l`** option (e.g. **`-l 48:3003`**) or guarantee that the clients follow some connection discipline.

Index

A

administrative application, VTCPD 89
administrative line 64
amu request debug 102
amu request status 102
application connection choice 59, 66
application-defined mode 29
application-host options 36
ASE
  VMS 34
  VTCPD 14, 18
asynchronous replies 61
asynchronous replies, VTCPD 89
attaching to VMS 34

B

backup LAN 38, 105

C

changing data length 59
clearing the connection 58
client mode 26
condition data, retrieving 92
conditions
  ertimeout 88, 96, 98, 106
  getfail 70, 82, 83, 106
  gotres 82, 83, 91, 94, 95, 106
  hostdown 70, 95, 106
  hostfail 70, 106
  hostup 70, 106
  inf 70, 106
  intertimeout 88, 97, 106
  invreq 106
  unexdata 89, 106
configuration files
  `services` 22–23
connection capacity 53
connection choice, application 66
connection options 35
connections overview
  connection options 19, 26
    application-defined mode 20, 29
    attaching to VMS 34
    UDP mode 20, 33
  multiple hosts 28
  multiple VTCPD 19, 28
  one connection 19, 28
  single host 19, 28
conventions
  manual 10

D

daemon backup 104
data
  receiving from a host, VTCPD 87
  receiving from a known ID, VTCPD 77
  retrieving condition data 92
  sending to a host, VTCPD 84
debugging
  VTCPD 102
delimiter, VTCPD messages 86
display message traffic 102

E

error messages 106
error timer 96
ertimeout 88, 96, 98, 106
examples
  obtaining the VTCPD port number 92
  VTCPD administrative application 89
exception conditions 106

F

fault tolerance 104
  backup LAN 105
  daemon backup 104
  monitoring host connections 104
  pinging the host from the VTCPD 104
  pinging the VTCPD from the host 105

G

getfail 70, 82, 83, 106
gotres 82, 83, 91, 94, 95, 106

H

handling exceptions 69
host communications 8
  TCP/IP 18
host connection
  freeing 95
  options 26
    application-defined mode 29
    attaching to VMS 34
    client mode 26

server mode 27
UDP mode 33
requesting dynamically 82
specifying 81
allowing VTCPD 82
from the application 81
host driven applications, VTCPD 72, 76
host sample code 104
host specifications
changing 79
establishing 79
hostdown 70, 95, 106
hostfail 70, 106
hostup 70, 106
hypertext links
types of 8
use of 8
See also manual: hypertext links in, using on line

I

inf 70, 106
intermediate resource timer 96
configuring 96
intertimeout 88, 97, 106
inverted mode 64
invreq 70
invreq 106

L

length header, VTCPD 86

M

manual
how to use 8
hypertext links in 8
intended audience for 8
organization of 9
scope of 8
using on line 8
message delimiter, VTCPD 86
message format options 48
message ID 50
message ID options 51
message ID, VTCPD
associating 78
sending data 84
message routing options 53
message timing, VTCPD 96
messages
VTCPD, multiple formats 91
monitoring host connections 38, 104
multiple formats 91
multiple VTCPDs, port number 91

N

name, VTCPD 22, 35, 91

P

PERIProducer applications 18
pinging the host from the VTCPD 104
pinging the VTCPD from the host 105
"Please hold on" message, customizing 97

Q

questions and answers 109
queuing requests 57

R

receiving message headers, VTCPD 88
registering the ID 64
releasing connection slot 69
replies routing 60
reply notification 61
reserving a slot 57
resource error timer 96
configuring 96

S

server mode 27
services file 22
simulate host input 102
simulating host using VTCPD 103
soliciting host data 73
status/exception conditions 106
status/exception conditions messages 106

T

TCP/IP connections 18, 19, 26, 27
time-outs 39
timers
intermediate resource 96
resource error 96

troubleshooting 107

U

UDP mode 33
UDP. *see* VAS/TCP daemon (VTCPD)
unexdata 62
`unexdata` 89, 106
unidentified host data 62
User Datagram Protocol (UPD). *see* VAS/TCP daemon (VTCPD)

V

variable length messages, VTCPD 86, 88
VAS/TCP daemon (VTCPD)
- administrative application 89
- associating the message ID 78
- asynchronous replies 89
- changing the length of a message 86
- debugging 102
- freeing the connection 95
- handling message timeouts 97
- host driven applications 72, 76
- host simulation 103
- host specifications
  - changing 79
  - establishing 79
- length header 86, 88
- limiting the interval for a reserved slot 75
- message ID 84
- message timing 96
- multiple VTCPDs 91
- port number 91
- receiving data 77, 78, 87
  - ignoring message timeouts 98
  - receiving headers 88
- reply notification 89
- requesting connections dynamically 82
- sending messages 84
- setting message timers 97
- specifying the connection 81
- status report 102

VOS
- processes 16–17

VTCPD 14, 18
VTCPD daemon names 22
VTCPD port assignments 22